Hard vs. Soft Bounds in Probabilistic Robustness Analysis And Generalized Source Coding and Optimal Web Layout Design

Thesis by

Xiaoyun Zhu

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2000

(Submitted May 16th, 2000)

To My Parents

Acknowledgements

I have been extremely fortunate to pursue my doctoral degree in such a dynamic and stimulating environment as Caltech, and to share my graduate years with such a group of talented, kind, and interesting people. First of all, I owe my deepest gratitude to my advisor, John Doyle, whose great vision and deep insights in science and engineering have always been a source of inspiration. My research has always been enlightened by his never ending effort in pursuing the big picture. Meanwhile, his devotion to athletic activities, and his caring attention to charity, reminded me about the other side of life.

Additionally, I have benefited from the help of many other professors in Caltech. I thank Richard Murray for giving me the opportunity to have hands-on experience with the helicopter experiment. This early experience offered me a chance to touch on the reality of controlling a physical system. I thank Mani Chandy for the discussions during our group meetings when I started working on the Web layout problem, for his little inspiring toy problems and his demonstration of the mathematically elegant way to treat them. Finally, I would like to acknowledge Michelle Effros and Thomas Hou, who along with John, Richard, and Mani, served on my thesis committee.

I extend my gratitude to my colleagues. To Yun Huang and Jie Yu, for the enjoyable collaboration on various topics of research, which contributed greatly to Chapter 4 and Chapter 13 of this thesis, respectively. To James Primbs, for numerous conversations on a broad range of subjects, and particularly, for his availability and willingness to help and his patience to correct my English in speaking and writing. To Sven Khatri and Pablo Parrilo, for collaboration and discussions on probabilistic robustness analysis resulting in Chapter 3 and Chapter 5. To Monica Ginnelli, my officemate, for kindly taking phone messages for me, and sharing thoughts on work and life in general.

Thanks also go to Charmaine, Shauna and Michelle, the secretaries of CDS department, for helping me resolve all sorts of little problems and tolerating such annoyances.

I dedicate my work to my parents. Their constant support and unconditional love have always been the most reliable force that I can depend on. From an early stage of my life, they set an example of how to be honest, hard working and warm-hearted. They provided me with a warm and happy place called home, and meanwhile, encouraged me to pursue things in the larger world.

I treasure my friends' help in the cause. Without their sustained friendship my life would be much emptier. I still vividly remember Xin An, Guangyang Wang, Sikun Lan and many others' help when I first came to the states. Without them I could not have overcome the roughness rather quickly. When I need help, there is one thing I am sure, Gang Qu and Zhe Jin would never turn me down, no matter how far away they are and how busy they are. And when I needed a ride during the dark carless days, Yong Wang always volunteered. I am fortunate to have my roommate, Jennifer Jiang, whose cooking I greatly enjoyed whenever I was too busy. Finally, I have been lucky to have Helen Si to share the happiness and roughness of life through my graduate years at Caltech.

Last but certainly not least, I thank my husband Qing, for loving me, supporting me, being patient with me, and most importantly, for showing me how to always keep faith in life no matter what it puts us through.

Abstract

Part I

The relationship between hard vs. soft bounds and probabilistic vs. worst-case problem formulations for robustness analysis has been a source of some apparent confusion in the control community, and this thesis attempts to clarify some of these issues. Essentially, worst-case analysis involves computing the maximum of a function which measures performance over some set of uncertainty. Probabilistic analysis assumes some distribution on the uncertainty and computes the resulting probability measure on performance. Exact computation in each case is intractable in general. In the past most research focused on computing hard bounds on worst-case performance. This thesis explores the use of both hard and soft bounds in probabilistic robustness analysis, and investigates the computational complexity of the problems through extensive numerical experimentation. We focus on the simplest possible problem formulations that we believe reveal the difficulties associated with more general probabilistic analysis.

By extending the standard structured singular value μ framework to allow for probabilistic descriptions of uncertainty, probabilistic μ is defined, which characterizes the probability distribution of some performance function. The computation of probabilistic μ involves approximating the level surface of the function in the parameter space, which is even more complex than the worst-case μ computation, a well-known NP-hard problem. In particular, providing sufficiently tight bounds in the tail of the distribution is extremely difficult. This thesis proposes three different methods for computing a hard upper bound on probabilistic μ, whose tightness can be tested by comparison with the soft bound provided by Monte-Carlo simulations. At the same time, the efficiency of the soft bounds can be significantly improved with the information from the hard bound computation. Among the three algorithms proposed, the LC-BNB algorithm is proven by numerical experiments to provide the best average performance on random examples. One particular example is shown in the end to demonstrate the effectiveness of the method.

Part II

The design of robust and reliable networks and network services has become an increasingly challenging task in today's Internet world. To achieve this goal, understanding the characteristics of Internet traffic plays a more and more critical role. Empirical studies of measured traffic traces have led to the wide recognition of self-similarity in network traffic. Moreover, a direct link has been established between the self-similar nature of measured aggregate network traffic and the underlying heavy-tailed distributions of the Web traffic at the source level.

This thesis provides a natural and plausible explanation for the origin of heavy tails in Web traffic by introducing a series of simplified models for optimal Web layout design with varying levels of realism and analytic tractability. The basic approach is to view the minimization of the average file download time as a generalization of standard source coding for data compression, but with the design of the Web layout rather than the codewords. The results, however, are quite different from standard source coding, as all assumptions produce power law distributions for a wide variety of user behavior models.

In addition, a simulation model of more complex Web site layouts is proposed, with more detailed hyperlinks and user behavior. The throughput of a Web site can be maximized by taking advantage of information on user access patterns and rearranging (splitting or merging) files on the Web site accordingly, with a constraint on available resources. A heuristic optimization on random graphs is formulated, with user navigation modeled as Markov Chains. Simulations on different classes of graphs as well as more realistic models with simple geometries in individual Web pages all produce power law tails in the resulting size distributions of the files transferred from the Web sites. This again verifies our conjecture that heavy-tailed distributions result naturally from the tradeoff between the design objective and limited resources, and suggests a methodology for aiding in the design of high-throughput Web sites.

Contents

List of Figures

List of Tables

Part I

Hard vs. Soft Bounds in Probabilistic Robustness Analysis

Chapter 1 Introduction

In control engineering accurate prediction of a system's behavior and successful control of the system are almost infeasible without the use of a model — a mathematical description of the underlying physical system that is being studied. However, no mathematical models are perfect, whether the model is derived from first principles, obtained using "black box" methodologies from empirical data, or a mixture of both. Possible sources that lead to discrepancy between the model and the real system include noise and disturbances from the environment where the system lives and operates, possible variations of the parameter values used in the model, or intentional employment of a simpler model on a highly complex system in order for the model to be analytically tractable and eventually usable in design and simulations. Examples of the latter would be the use of a lower order model which neglects higher order dynamics, or the use of a linear model as an approximation for a nonlinear system. The above factors are considered as "uncertainty" associated with the corresponding model. Control design without taking into account model uncertainty can lead to instability or performance deterioration, which may sometimes result in catastrophic system failure. The subject of *robust control* involves the development of systematic tools for dealing with model uncertainty in order to reduce the risk of system failure as much as possible and achieve high performance specifications even in the face of uncertainty. These tools can be coarsely divided into the following three categories:

Uncertainty modeling: A mathematical infrastructure for quantification of uncertainty for it to be incorporated into system analysis and design. This essentially involves mathematical descriptions of a "set" of models instead of a single model, which allows for a more complete characterization of a physical system.

Robustness analysis: Theories and computational tools developed to predict the behavior of the model when the description of the uncertainty has been provided.

Robust control synthesis: Controller design methodologies that guarantee stability and performance for the set of models that describe the underlying system.

The focus of this thesis is on robustness analysis, and in particular, probabilistic robustness analysis.

1.1 Background

1.1.1 Worst-Case vs. Probabilistic Robustness Analysis

In robustness analysis there are essentially two problem formulations: worst-case robustness analysis and probabilistic robustness analysis. In worst-case analysis, the major concern is the worst outcome for a system in the presence of uncertainty. Mathematically it involves finding the maximum of a function which measures the system's performance over some set of parameters or operators that describes the uncertainty. In probabilistic analysis, the question of interest is what is the probability of bad performance. To answer this question, a probabilistic description of the uncertainty is required, then the resulting probability measure on the performance can be evaluated.

Much effort in the history of worst-case robustness analysis has been devoted to characterizing conditions of guaranteed stability and performance for uncertain systems that are described by a set of models. The small gain theorem introduced by Zames [77] in 1965 was the first significant milestone in addressing this issue. It states that an interconnected system is guaranteed to be stable if both systems in the loop are stable and the loop gain is less than one, which provided an exact robust stability test with respect to unstructured dynamic uncertainty. However, a set of models using unstructured uncertainty could be much larger than necessary to cover the real system. The introduction of structured or block diagonal uncertainty and the structured singular value μ by Doyle [24] and Safonov [64] in 1982 provided a much less conservative framework for analyzing systems with various types of uncertainties. In particular, the usefulness and flexibility of the μ framework as a robustness analysis tool are due to the fact that many robust stability and performance problems can be recast as μ computation problems with appropriate types of uncertainty structure.

On the other hand, a worst-case paradigm can be overly conservative for many practical applications. When applying this framework to robust control design, the resources available will be used to guard against some worst-case event, which may only happen with extremely small probability that can be neglected for engineering purposes. In general, the probability distribution of a system's performance largely depends on the distribution of the uncertainty that comes into play. Suppose the uncertainty consists of a set of parameters. If it is known a priori that some parameter values are highly unlikely compared with others, it seems unwise to treat them equally rather than to take advantage of this information in stability and performance analysis. Although a probabilistic framework may not always be appropriate, it does arise naturally in some engineering practices. For example, estimating the yield of chips as the result of some manufacturing process, or assessing the risk of failures for some complex engineering systems such as chemical process facilities and space systems, based on probabilities of component failures [50]. In these situations, probability distributions on the uncertainty has natural interpretations or can be obtained through a combination of expert opinion and empirical data [2]. Other work in choosing probability distributions for uncertainty include the study of the worst-case distribution of uncertain parameters by Barmish [10]. Once the probability distribution on the uncertainty is given, the problem becomes how to compute the resulting probability distribution on the system's performance.

In general, exact computation for robustness analysis, either worst-case or probabilistic, is intractable. It is quite easy to see intuitively why this might be so, even for linear systems with parametric uncertainty. Suppose that for fixed parameters we can compute some function f to obtain a measure of performance and the cost to do so is C, and that we have n uncertain parameters. Suppose that we want to compute f for r values in each parameter in order either to estimate the maximum (worst-case) or average (probabilistic) performance, or to estimate the probability that f exceeds some threshold. The total number of possible combinations of parameter values is then r^n and the cost of all the evaluations is Cr^n. Thus the growth rate in

n is exponential, which means that the addition of even a few parameters to a model can cause severe increases in computation. This intuition is supported by theoretical results that show that even for linear models with parametric uncertainty, evaluating virtually any robustness measure is NP hard in the number of parameters [12]. Although the computational implication of NP-hardness is still a fundamentally open question in the field of computational complexity, saying that a problem is NP-hard is generally taken to mean that the problem cannot be computed in polynomial time in the number of parameters in its worst case. It is important to note that being NP-hard is a property of the problem itself, not any particular algorithm. The fact that μ computation is NP-hard strongly suggests that given *any* algorithm, there will be problems for which the algorithm cannot find μ in polynomial time. In fact, even the approximation problem for μ is NP-hard [31]. That is, computing μ to within a constant factor of the optimum is NP-hard. So it is a "hard" NP-hard problem from the point of view of computational complexity. If anything, a probabilistic framework makes this even more difficult since the computation is more involved. Efficient computational schemes are needed to deal with this complexity.

1.1.2 The Monte-Carlo Approach

There are several approaches to overcoming the apparent intractability of worst-case or probabilistic robustness analysis. An indirect approach which is well-known and has been the industry standard for decades is so-called Monte-Carlo simulation [63, 69]. Suppose the joint probability distribution of the uncertain parameters p_i, $i = 1, \ldots, n$ is given. Figure 1.1 shows the special case where p_i are independently distributed, which may not be true in general. To experimentally estimate the distribution on the performance function f, one can compute f at random samples of p_i generated according to its distribution. This can then be subjected to standard statistical tests like any experimental data to produce "soft" bounds through hypothesis tests. The beauty of the Monte-Carlo approach is that the accuracy of the estimates trivially does not depend on the dimension of the parameter space, so there is no

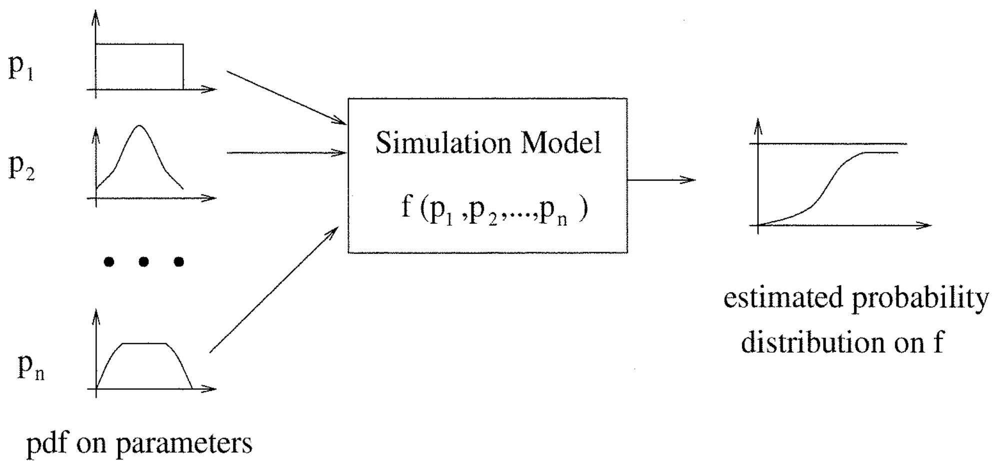
p_1
p_2
p_n
pdf on parameters
Simulation Model
$f(p_1, p_2, \ldots, p_n)$
estimated probability distribution on f

Figure 1.1: Monte-Carlo simulation.

growth whatsoever in the computation cost with the number of parameters [17]. The only cost is that of the function evaluation itself and the number of times it must be repeated to obtain a statistically significant sample size. Furthermore, Monte-Carlo can be applied to any simulation or experiment, so it is applicable to many problems that lack the mathematical structure for more systematic analysis.

The main difficulty with the Monte-Carlo soft bounds approach is it does not actually compute the probability distribution of performance, but only indirectly assesses it. That is, we do not obtain hard bounds like "the model achieves the desired performance 99% of the time," but instead we obtain soft bounds like "we can be 95% confident based on the experimental data that the model achieves the desired performance 99% of the time." What this means more precisely is that a model that has acceptable performance for 99% of the assumed uncertainty set would produce data as good as what we have observed for 95% of repeated experiments. The actual probability distributions remain unknown, and would require the prohibitive computation of multidimensional integrals with the inherent intractability described in the previous subsection, so it is still possible that the true probabilities are much worse than 99%. The need to use such confidence levels can be particularly annoying when they are not naturally motivated or when estimating the probability of rare events with high confidence levels, which requires an enormous number of Monte-Carlo trials.

On the other hand, we may simply want to know if anything bad can happen for some set of parameters, and there is no natural way to interpret the probability distribution of the parameters or the resulting probability distribution of the performance. In this case, a worst-case problem formulation may be more appropriate. While it is possible, in principle, to use Monte-Carlo to estimate soft bounds on worst-case performance, this is much more awkward than its conventional use and is unlikely to be practical.

1.1.3 Hard Bounds vs. Soft Bounds

There are alternatives to the Monte-Carlo technique that provide different and complementary answers, and the development of such alternatives has been a driving force behind much research in robust control theory for the last twenty years. The difficulty is that if we want to avoid soft estimates, we must overcome the worst case intractability implied by the NP-hardness of our problems. The approach that has proven to be most successful involves computing hard bounds, as opposed to the soft bounds provided by Monte-Carlo, and refining the bounds by using branch and bound (B&B) techniques. A good example is the development of computationally tractable upper and lower bounds for μ and many efficient algorithms to compute and improve them [30, 75, 11, 76, 53, 54, 52].

The hard bounds approach has been primarily used to compute worst-case performance, but it could in principle be used to compute hard bounds on probabilistic measures as well. Similarly, Monte-Carlo could be used to obtain soft bounds on worst-case performance, although it would require possibly unverifiable assumptions on the models. Nevertheless, it is somewhat artificial to exclusively associate Monte-Carlo and probabilistic problem formulations, although historically most research has focused on computing hard bounds for worst case, and little attention has been focused on computing hard bounds for probabilistic measures. However, as was mentioned previously, the Monte-Carlo simulation becomes extremely inefficient when it is used to assess the probability of rare events, because a large number of samples will be wasted on benign events. In this case it is desirable to have some kind of hard bounds on the tail of the distribution.

The big advantage of hard bounds for either worst-case or probabilistic is that they are guaranteed and can be refined by branch and bound. The potential difficulty is that there exist examples for which this refinement may take prohibitively long. For many worst-case problems, it seems that such examples are extremely rare to the point that it is unlikely that anyone would ever encounter one without specifically constructing it. It is interesting to note that this is true of course with essentially all numerical algorithms, even those we think of as polynomial time, such as eigenvalue and singular value computation. For these problems, however, there is a much clearer picture of the nature of "hard" problems, whereas for the problems in robustness analysis, the evidence is entirely numerical.

The problem of finding hard bounds on probability distributions has received almost no attention in the robust control literature. Since it involves essentially bounding the integral of some complex function over some parameter space, it appears harder than the corresponding worst-case problem of computing the maximum value of the function. Investigation of this problem is the focus of this thesis.

1.2 Outline of Part I

The main contribution of this thesis is that it is the first attempt to develop computable hard bounds for probabilistic robustness analysis as a complement to soft bounds provided by traditional statistical methods, such as Monte-Carlo simulations. Probabilistic μ is introduced as an extension of the structured singular value μ from the worst-case to the probabilistic framework. We explore the possibility of applying branch and bound techniques in computing hard upper bounds and improving soft bounds on probabilistic μ and investigate the technical difficulty and computational challenges associated with the problem. In addition, more intelligent computational schemes are developed accordingly to cope with the complexity of the problem.

The thesis starts with a review in Chapter 2 of the structured singular value μ

in its standard framework, which is a useful tool for analyzing robust stability and robust performance from a worst-case perspective. Computationally tractable upper and lower bounds for μ are described. In addition, purely real μ and implicit μ are discussed which will be used in later chapters.

In Chapter 3 the standard μ framework is extended to allow probabilistic uncertainty descriptions with which probabilistic μ is defined. Further simplification of the description leads to the definition of purely probabilistic μ, which corresponds to the complementary cumulative distribution of a performance function. The relationship between standard μ as a worst-case measure and purely probabilistic μ as a distribution leads to the possibility of using a combination of the standard μ upper bound and a branch and bound algorithm to compute hard upper bounds and soft bounds on probabilistic μ. The rest of the thesis is devoted to the development of three different algorithms for implementing this idea. Their performance is tested through extensive numerical experimentation and an overall comparison will be provided at the end of this thesis.

Chapter 4 briefly reviews branch and bound as a general optimization technique and demonstrates through numerical experiments how a naive branch and bound algorithm can effectively improve the upper and lower bound computation for standard μ, but fails to obtain sufficiently tight bounds for purely probabilistic μ. Analysis of two special cases reveals the fact that the difficulty in the computation is also present with rank-one problems, for which exact bounds are available in the worst-case computation. Intuition into rank-one problems motivated the study of μ with richer classes of uncertainty descriptions, including spherical μ and elliptical μ [39, 58] introduced by Khatri and Parrilo and μ with linear cuts [82] discussed in the next chapter.

μ with linear cuts is defined when linear constraints are included to describe the real parametric uncertainty in the μ formulation. Three methods for computing upper bounds for μ with linear cuts are described in Chapter 5, including the elliptical method that employs the elliptical μ upper bound, the implicit method that implements the linear constraint through an implicit system, and the parallelogram method that augments the uncertainty set so that the standard μ upper bound can be used. These three methods are compared though numerical experiments, which rank the implicit method above the other two in terms of average performance.

Using the results in Chapter 5, a probabilistic μ upper bound using linear cuts is presented in Chapter 6. When applied to rank-one problems or problems that are close to rank-one, the linear cut algorithm either provides the exact bound or outperforms the naive branch and bound algorithm with axially aligned cuts. However, its performance on general random matrices is unsatisfactory, which motivated the search for more intelligent algorithms to deal with general random matrices.

The last algorithm presented in Chapter 7 is a mixture of the naive branch and bound algorithm and the linear cut algorithm. It combines the strengths of both algorithms so that better bounds on probabilistic μ can be computed. Numerical experiment results verify that the average performance of the last algorithm is much greater than the previous two algorithms. The advantage of this algorithm compared to a standard Monte-Carlo simulation algorithm is also demonstrated through a particular example.

Finally, some concluding remarks and suggestions for future research are given in Chapter 8.

1.3 Notation

The notation in this thesis is fairly standard. If M is a matrix, then M^T, M^* denote the transpose and complex conjugate transpose of M, respectively. The same convention applies to a vector x. $Ker(M)$ is the kernel of M. The $n \times n$ identity matrix will be denoted by I_n, I is used when the dimension is obvious. Zero matrices will be denoted by 0. $diag[A_1, \cdots, A_n]$ is a block diagonal matrix with A_i's on the diagonal.

The maximum singular value of a matrix M is denoted by $\overline{\sigma}(M)$. For a square matrix M, $\rho(M)$ and $\rho_r(M)$ denote the spectral radius and the real spectral radius, respectively, while $\overline{\lambda}_r(M)$ denotes the maximum positive real eigenvalue. When M has no (positive) real eigenvalues, $\rho_r(M)$ $(\overline{\lambda}_r(M\Delta)) = 0$. $Tr(M)$ is the trace of M, and $det(M)$ is the determinant of M.

For $x \in \mathbb{C}^n$, $||x||$ denotes the vector 2-norm, while for a matrix $M \in \mathbb{C}^{n \times n}$, $||M||$ stands for the induced 2-norm, e.g., $||M|| = \overline{\sigma}(M)$.

The Hadamard (or Schur) element-wise product of two matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ of the same dimensions is defined as $A \circ B \equiv [a_{ij} b_{ij}]$. It will be used later to specify particular matrix structures.

For an uncertainty structure $\boldsymbol{\Delta}$, $\boldsymbol{B\Delta}: = \{\Delta \in \boldsymbol{\Delta} : ||\Delta|| \leq 1\}$. $\boldsymbol{B}(\Delta_0, r)$ denotes the axially aligned hyperrectangle centered at Δ_0, and the vector r contains the half lengths of the sides. The volume of a region $\boldsymbol{R} \in \mathbb{R}^n$ is denoted by $V(\boldsymbol{R})$. For two sets $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ in $\mathbb{R}^n$, $\boldsymbol{R}_1 \backslash \boldsymbol{R}_2$ denotes the set $\{\Delta : \Delta \in \boldsymbol{R}_1,\ \Delta \notin \boldsymbol{R}_2\}$.

Chapter 2 The μ Paradigm

This chapter reviews the definition of the structured singular value μ, its relationship with Linear Fractional Transformations (LFTs), and it upper and lower bound computation. A special case that only involves real parametric uncertainty is discussed. In the end an extension of μ to implicit systems is described.

2.1 The General Framework

The use of μ in the study of robust stability and performance is motivated by its close relationship with Linear Fractional Transformations (LFTs), a general class of linear feedback loops used to define the set of models associated with the uncertain system. See [56] for an in-depth discussion on LFTs.

2.1.1 Linear Fractional Transformations

Let $M \in \mathbb{C}^{(n+m)\times(n+m)}$ and $\Delta \in \mathbb{C}^{n\times n}$. M can be partitioned as $M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$, with $M_{11} \in \mathbb{C}^{n\times n}$ and $M_{22} \in \mathbb{C}^{m\times m}$. Then the block diagram in Figure 2.1 defines an LFT, denoted by $\Delta \star M$.

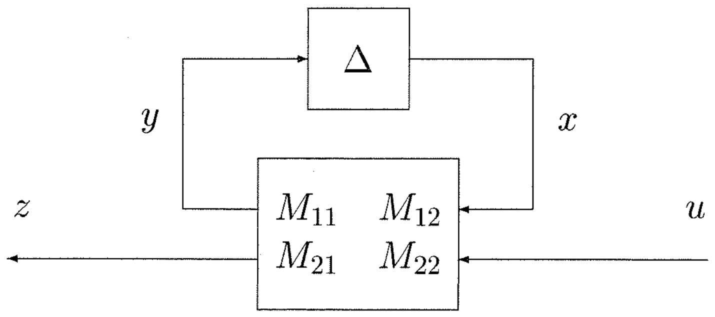

Figure 2.1: Linear Fractional Transformation.

We say that the LFT $\Delta \star M$ is *well posed* if and only if there exists a unique solution to the following loop equations:

$$\begin{aligned} y &= M_{11}x + M_{12}u \\ z &= M_{21}x + M_{22}u \\ x &= \Delta y. \end{aligned} \tag{2.1}$$

When the LFT is well posed, it uniquely defines a mapping from $u \rightarrow z$, i.e., $z = (\Delta \star M)u$, where

$$\Delta \star M \colon = M_{22} + M_{21}\Delta(I - M_{11}\Delta)^{-1}M_{12}. \tag{2.2}$$

From this formulation, it is easy to see that the LFT $\Delta \star M$ is well posed if and only if the matrix $I - M_{11}\Delta$ is nonsingular. If $I - M_{11}\Delta$ is singular, there are infinitely many solutions to (2.1), in which case $||x||$ and $||y||$ could be arbitrarily large. Then the system is in some sense unstable.

The above LFT defined is called an *upper* LFT. Similarly, a *lower* LFT $M \star \Delta$ can be defined as

$$M \star \Delta \colon = M_{11} + M_{12}\Delta(I - M_{22}\Delta)^{-1}M_{21}. \tag{2.3}$$

LFTs are a natural generalization of state space representations of linear time invariant (LTI) systems. When $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, and $\Delta = \frac{1}{z}I$,

$$\Delta \star M = D + C(zI - A)^{-1}B \tag{2.4}$$

becomes the transfer function of a discrete-time system. By including various types of uncertainty operators into Δ, the LFT system defined in Figure 2.1 can represent a fairly general class of uncertain systems, for which robust stability and performance can be analyzed by studying the well posedness of the LFT with respect to appropriate set of uncertainties.

2.1.2 Definition of μ

If the use of LFTs in modeling uncertain systems laid down a basis for robustness analysis, then the introduction of the structured singular value μ provided a useful tool for directly evaluating stability and performance in the set of models described by the LFT.

As can be seen in the previous section, the well posedness of the LFT system $\Delta \star M$ completely depends on the feedback interconnection between Δ and M_{11}. This interconnection is pulled out of Figure 2.1 and shown in Figure 2.2, where M_{11} is renamed as M for simplicity. Δ represents the uncertainty in the system.

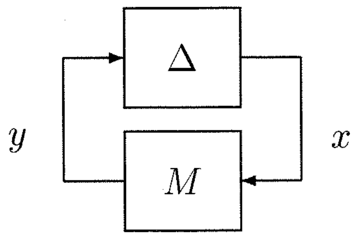

Figure 2.2: Standard Interconnected System (S1).

The structured singular value μ addresses the existence of nontrivial solutions to the loop equations

$$\begin{cases} y = Mx \\ x = \Delta y. \end{cases} \tag{2.5}$$

It is easy to see that there are no nontrivial solutions to (2.5) if and only if the matrix $I - M\Delta$ is nonsingular, which is equivalent to the well posedness of the LFT defined in Figure 2.1 when $M = M_{11}$.

The uncertainty matrix Δ has a block diagonal structure in general. The kinds of blocks included depends on the types of uncertainties and performance objectives for each specific problem. The general μ framework accommodates real parameter variations, change of frequencies, and unmodeled dynamics, which are represented by repeated real scalars, repeated complex scalars, and full complex blocks, respectively.

Let m_r, m_c and m_C be three nonnegative integers ($m = m_r + m_c + m_C \leq n$) that specify the number of uncertainty blocks of each type. Then let the m-tuple of positive integers

$$\mathcal{K} = (k_1, \ldots, k_{m_r}, k_{m_r+1}, \ldots, k_{m_r+m_c}, k_{m_r+m_c+1}, \ldots, k_m) \tag{2.6}$$

correspond to the dimensions of individual blocks, with $\sum_{i=1}^{m} k_i = n$. The set of allowable uncertainties is defined as follows:

$$\begin{aligned}\boldsymbol{\Delta} :&= \{diag[\delta_1^r I_{k_1}, \ldots, \delta_{m_r}^r I_{k_{m_r}}, \delta_1^c I_{k_{m_r+1}}, \ldots, \delta_{mc}^c I_{k_{m_r+m_c}}, \Delta_1^C, \ldots, \Delta_{mC}^C] : \\ &\delta_i^r \in \mathbb{R},\ \delta_i^c \in \mathbb{C},\ \Delta_i^C \in \mathbb{C}^{k_{m_r+m_c+i} \times k_{m_r+m_c+i}}\}.\end{aligned} \tag{2.7}$$

For a given perturbation $\Delta \in \boldsymbol{\Delta}$, its "size" is measured by the ∞-norm $\|\Delta\| = \sigma(\Delta)$. Accordingly, $\boldsymbol{B\Delta}$ is the unit-norm bounded set in $\boldsymbol{\Delta}$.

The definition of the structured singular value μ is a generalization of the singular value and the spectral radius for matrices.

Definition 2.1 *For $M \in \mathbb{C}^{n \times n}$, the structured singular value $\mu_{\boldsymbol{\Delta}}(M)$ with respect to a given block structure $\boldsymbol{\Delta}$ is defined as*

$$\mu_{\boldsymbol{\Delta}}(M) := \frac{1}{\min\{\overline{\sigma}(\Delta) : \Delta \in \boldsymbol{\Delta}, \det(I - M\Delta) = 0\}} \tag{2.8}$$

unless $det(I - M\Delta) \neq 0, \forall \Delta \in \boldsymbol{\Delta}$, in which case $\mu_{\boldsymbol{\Delta}}(M) := 0$.

In the other words, $\frac{1}{\mu_{\boldsymbol{\Delta}}(M)}$ measures the "size" of the smallest perturbation in the space of allowable Δ which makes the matrix $I - M\Delta$ singular. As a consequence, the following theorem holds.

Theorem 2.2 (Well Posedness [78]) *The LFT $\Delta \star M$ is well posed for all $\Delta \in \boldsymbol{B\Delta}$ if and only if $\mu_{\boldsymbol{\Delta}}(M_{11}) < 1$.*

The above theorem states that μ provides an exact test for robust stability. Using μ for robust performance analysis is based on the following theorem.

Given two uncertainty structures $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$, define a new structure

$$\mathbf{\Delta} : = \left\{ \begin{bmatrix} \Delta_1 & \\ & \Delta_2 \end{bmatrix} : \Delta_1 \in \mathbf{\Delta}_1,\ \Delta_2 \in \mathbf{\Delta}_2 \right\}. \tag{2.9}$$

Theorem 2.3 (Main Loop Theorem [78])

$$\mu_{\mathbf{\Delta}}(M) < 1 \iff \begin{cases} \mu_{\mathbf{\Delta}_1}(M_{11}) < 1 \\ \sup_{\Delta_1 \in \boldsymbol{B\Delta}_1} \mu_{\mathbf{\Delta}_2}(\Delta_1 \star M) < 1. \end{cases}$$

The main loop theorem enables the robust performance test to be recast as a μ computation problem, with $\mathbf{\Delta}_1$ representing all the uncertainties, and $\mathbf{\Delta}_2$ reflecting the performance objective. For details and examples see [78].

2.1.3 μ Upper Bound and Lower Bound

The μ problem with the uncertainty structure $\mathbf{\Delta}$ defined in (2.7) is referred to as the "mixed μ" problem, because $\mathbf{\Delta}$ contains both real and complex uncertainty blocks. It has been proven that computation of the mixed μ is NP-hard [12]. Fortunately, there are upper and lower bounds for μ with polynomial time algorithms.

The lower bound is based on the following alternative definition of μ:

$$\mu_{\mathbf{\Delta}}(M) = \sup_{\Delta \in \boldsymbol{B\Delta}} \rho_r(M\Delta) = \sup_{\Delta \in \boldsymbol{B\Delta}} \overline{\lambda}_r(M\Delta) \tag{2.10}$$

The above optimization problem is not convex, so there are no guarantees of finding the global optimum. A power iteration algorithm can be used to search for the optimum [75, 53]. Although the iteration is not not guaranteed to converge, and even if it converges, it may converge to a local maximum, it is faster and has better global convergence than standard optimization.

On the other hand, an upper bound $\overline{\mu}_{\mathbf{\Delta}}(M)$ can be constructed using Linear

Matrix Inequalities (LMIs) [30]:

$$\overline{\mu}_{\boldsymbol{\Delta}}(M) = \inf_{D \in \boldsymbol{D}, G \in \boldsymbol{G}} \{\beta > 0 : M^* D M + j(GM - M^* G) - \beta^2 D < 0\} \tag{2.11}$$

where

$$\begin{aligned} \boldsymbol{D}\colon = \{&diag[D_1, \dots, D_{m_r+m_c}, d_1 I_{k_{m_r+m_c+1}}, \dots, d_{M_C} I_{k_m}] : \\ &0 < D_i = D_i^* \in \mathbb{C}^{k_i \times k_i},\ d_i > 0\}, \end{aligned} \tag{2.12}$$

and

$$\boldsymbol{G}\colon = \{diag[G_1, \dots, G_{m_r}, 0, \dots, 0] :\ G_i = G_i^* \in \mathbb{C}^{k_i \times k_i}\}. \tag{2.13}$$

The upper bound in (2.11) is a generalized eigenvalue problem (GEVP), which can be solved using an interior point method for LMIs [51, 11]. D and G are called "scaling matrices," which are used to exploit the block diagonal structure of Δ and the phase information in the real parameters, respectively. The optimization is convex, but the optimum does not achieve μ in general, and there can be large gaps between the optimal upper bounds and the real values of μ.

Based on the above discussions, we can see that although both upper and lower bounds are computationally tractable, they are not guaranteed to be tight because there exist problems where either the lower bound or the upper bound can be poor. This is the motivation for using branch and bound techniques to refine the bounds, which will be reviewed in Chapter 4.

2.2 Purely Real μ

The mixed μ reviewed in the previous section is the most standard μ formulation, which accounts for both real parameter variations and unmodeled dynamics. This formulation will not be used in this thesis. Instead, we will focus on a much simpler setting, where only real parametric uncertainty is considered. The reason will become clear after the notion of probabilistic μ is introduced later in this chapter. It is also assumed that all the real parameters are nonrepeated to further simplify the notation. Hence, the uncertainty structure that will be used throughout this thesis is

$$\mathbf{\Delta}: = \{diag\,[\delta_1, \dots, \delta_n] : \ \delta_i \in \mathbb{R}\}. \tag{2.14}$$

The structured singular value μ defined with the uncertainty structure in (2.14) is a special case of the one in Definition 2.1. It is referred to as the "purely real μ," or "μ on a box." The second name comes from the fact that the set $\boldsymbol{B\Delta} = \{\Delta \in \mathbf{\Delta} : ||\Delta|| \leq 1\}$ in this case is the the unit box in $\mathbb{R}^n$.

Similar to the case of mixed μ, computing purely real μ is also NP-hard [12]. The lower bound for purely real μ involves a non-convex optimization of $\overline{\lambda}_r(M\Delta)$ on the unit box $\boldsymbol{B\Delta}$, based on (2.10). And there is no gap between its optimum and $\mu_{\mathbf{\Delta}}(M)$. Here we will take a step further and assume that M is real. The class of μ problems with real M is encountered when M is constructed from state space representations of systems using 'A, B, C, D' matrices. The following lemma is taken from [74].

Lemma 2.4 *For a real matrix $M \in \mathbb{R}^{n\times n}$ and an uncertainty structure $\mathbf{\Delta}$ defined in (2.14), it suffices to consider perturbations at the vertices of the allowed perturbation set.*

The above lemma says that in the special case defined above, the maximum of the optimization in (2.10) must be achieved on the vertices of the unit box. Therefore, μ can be computed exactly by checking a finite number of points. However, the required computation grows exponentially with problem size, which is consistent with the NP-hardness nature of the problem. To illustrate how fast this growth can be, Table 2.1 shows the computation time needed to check $\overline{\lambda}_r(M\Delta)$ on all the 2^n vertices of $\boldsymbol{B\Delta}$ versus problem size n. Times are estimates for a SUN Ultra Sparc workstation.

Obviously, even though the computation seems trivial, the growth rate of the time required is devastating for large problems. Again instead of computing μ directly, lower bound and upper bound of μ are computed using polynomial time algorithms.

Problem Size (n)				
4	8	16	32	64
0.01 sec	0.3 sec	5 min	3 years	9×10^{10} years

Table 2.1: Time estimates for exact computation of purely real μ.

The upper bound for the mixed μ in (2.11) is reduced to

$$\overline{\mu}_{\boldsymbol{\Delta}}(M) = \inf_{D \in \boldsymbol{D}} \{\beta > 0 \colon M^T D M - \beta^2 D < 0\} \tag{2.15}$$

where

$$\boldsymbol{D} = \{diag[d1, d2, \dots, d_n] : \ d_i > 0\}. \tag{2.16}$$

Note that the G scaling is not necessary here since M is real and δ_i are nonrepeated [74].

Suppose $\overline{\mu}_{\boldsymbol{\Delta}}(M) = \beta^*$, then it is guaranteed that the matrix $I - M\Delta$ is nonsingular for all $\Delta \in \frac{1}{\beta^*}\boldsymbol{B\Delta}$. So the search for the minimum β is essentially a scaling on $\boldsymbol{B\Delta}$ to find the biggest box in $\mathbb{R}^n$ in which $I - M\Delta$ is invertible.

When $\boldsymbol{B\Delta}$ is only scaled along the directions of m parameters ($m < n$), while the sizes of other $n - m$ parameters remain fixed, the skew version of purely real μ can be defined. In this case, with appropriate ordering of the parameters, the upper bound is identical to the one in (2.15) with β^2 replaced by $\begin{bmatrix} I_{n-m} & \\ & \beta^2 I_m \end{bmatrix}$. This upper bound will be used later for μ with linear cuts in Chapter 6.

Skew μ can be generalized to the mixed μ case, where it is typically used for the computation of robust performance. The idea is to fix the size of the uncertainty block while changing the scale of the performance block so that the guaranteed level of performance can be found.

2.3 Implicit μ

The above μ framework can be extended to implicit systems. An implicit framework allows one to add algebraic constraints to the standard interconnected system (S1), as shown in Figure 2.3.

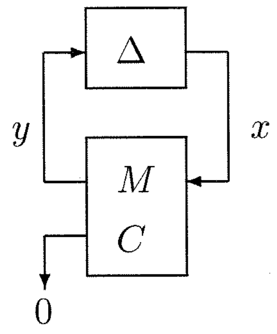

Figure 2.3: Standard Implicit System (S2).

Implicit μ addresses the existence of nontrivial solutions to the loop equations defined in Figure 2.3, with the constraint $Cx = 0$ posed on the output of Δ. The definition of implicit μ and its upper bound given in [57] (simplified for the purely real case) follow. The implicit μ upper bound will be used in later chapters.

Definition 2.5 *For the implicit system (S2), the structured singular value $\mu_{\mathbf{\Delta}}(C, M)$ is defined as*

$$\mu_{\mathbf{\Delta}}(C,M) :\, = \frac{1}{\min\{\overline{\sigma}(\Delta) :\ \Delta \in \mathbf{\Delta},\ Ker \begin{bmatrix} I - \Delta M \\ C \end{bmatrix} \neq 0\}}, \qquad (2.17)$$

unless $Ker \begin{bmatrix} I - \Delta M \\ C \end{bmatrix} = 0, \forall \Delta \in \mathbf{\Delta}$*, in which case* $\mu_{\mathbf{\Delta}}(C,M) :\, = 0.$

Let $C_\perp$ be the matrix whose columns form a basis for $Ker(C)$. Then the following is an upper bound on $\mu_{\mathbf{\Delta}}(C, M)$.

$$\overline{\mu}_{\mathbf{\Delta}}(C,M) = \inf_{D \in \boldsymbol{D}} \{\beta > 0 :\ C_\perp^*(M^*DM - \beta^2 D)C_\perp < 0\}. \qquad (2.18)$$

Chapter 3 Probabilistic μ

The standard μ framework reviewed in the previous chapter is essentially a worst-case paradigm. The computation of μ is equivalent to the search for the largest set of perturbations so that stability or performance criterion is satisfied by the system with all possible perturbations in the set. This approach can be overly conservative in situations where the above set can be greatly enlarged at the expense of a small risk, especially when a probabilistic description of the uncertainty is appropriate and available. It appears natural to use this information in assessing the probability of instability or poor performance, which can help setting tradeoffs when design decisions are made. This chapter extends the standard μ framework to include probabilistic descriptions for the uncertainty Δ in the interconnected system (2.2). This new framework is referred to as the probabilistic μ framework, to be distinguished from the standard μ framework which only deals with worst-case robustness problems.

3.1 The General Framework

An early investigation of probabilistic μ was done in [81], where $\overline{\lambda}_r(M\Delta)$ was chosen as the performance measure of the system with perturbation Δ. Assume that the distribution of Δ is given. The objective is to compute the resulting complementary cumulative distribution of the performance function:

$$P(\gamma) = P[\overline{\lambda}_r(M\Delta) \geq \gamma]. \tag{3.1}$$

A more general framework for probabilistic μ was described in [38], where the uncertainty Δ consists of two parts: $\Delta = \begin{bmatrix} \Delta_p & 0 \\ 0 & \Delta_w \end{bmatrix}$, where Δ_p is the part of the uncertainty that admits a probabilistic description, and Δ_w is the rest where only the worst-case is concerned. The most natural way to think of this separation is Δ_p contains the real parametric uncertainty blocks, while Δ_w consists of the stability and performance blocks. For example, if $\Delta_w = z^{-1}I$ for a discrete-time system, the probability of the system achieving stability is given by

$$P[\mu_{\mathbf{\Delta}_w}(M \star \Delta_p) < 1]. \tag{3.2}$$

And if $\Delta_w = \begin{bmatrix} z^{-1}I & 0 \\ 0 & \Delta_f \end{bmatrix}$, where Δ_f is a complex full block and $\|\Delta_f\| \leq \gamma$, then the probability of the system being stable and achieving the performance level γ is given by

$$P\left[\mu_{\mathbf{\Delta}_w}\left(\begin{bmatrix} I & 0 \\ 0 & \frac{1}{\gamma}I \end{bmatrix} M \star \Delta_p\right) < 1\right]. \tag{3.3}$$

Monte-Carlo simulation can be used to estimate the above probabilities. The difficulty is, each sample trial involves a μ calculation, which is NP-hard itself. When a large number of samples are required, the overall computation is hardly tractable. For special classes of problems where Δ_w has rather simple structure, like in the case of (3.2) or (3.3), μ equals its upper bound, then the computation become more feasible. However, it is not clear yet how to compute hard bounds on these probabilities. As a starting point, we will focus in this thesis on a much simplified case: the purely probabilistic case, which is discussed in the next section.

3.2 Purely Probabilistic μ

In the case where the uncertainty description is purely probabilistic, $\Delta = \Delta_p$. Since it is hard to interpret probability distributions on unmodeled dynamics, we would further assume that Δ contains only real parametric uncertainties, i.e., $\Delta \in \mathbf{\Delta}$ where $\mathbf{\Delta}$ is defined in (2.14). Moreover, in most practical applications the uncertain parameters vary in a bounded set. Without loss of generality, it is always assumed that

$\Delta \in \boldsymbol{B\Delta}$ because other cases can be easily normalized by appropriate scaling on rows and columns of M. To define probabilistic μ in this purely real case, we need to first define a μ-type function to be consistent with the general setting. Considering that $\mu_{\boldsymbol{\Delta}}(M) = \max_{\Delta \in \boldsymbol{B\Delta}} \overline{\lambda}_r(M\Delta)$, it is natural to choose $\overline{\lambda}_r(M\Delta)$ as in [81]. Later discussion shows that this choice happens to be consistent with the notion of "stability" for this purely real case.

Definition 3.1 *The purely probabilistic μ is defined as*

$$\mu_{\boldsymbol{\Delta}}^{P}(M,\alpha) = \gamma \qquad \textit{if} \qquad P[\overline{\lambda}_r(M\Delta) \geq \gamma] = \alpha.$$

The superscript 'P' emphasizes the probabilistic setting. In addition to specifying the matrix M and the associated uncertainty structure $\boldsymbol{\Delta}$ as in the standard μ case, one more parameter is needed to define probabilistic μ — the risk level α. The relationship between α and γ is demonstrated in Figure 3.1. Suppose the function $\overline{\lambda}_r(M\Delta)$ provides some inverse performance measure of the system, i.e., the higher the function, the worse the performance, then γ is a variable for all possible performance levels, with $\mu_{\boldsymbol{\Delta}}(M)$ as its worst case. $\alpha = P(\gamma)$, where $P(\gamma)$ is defined in (3.1), corresponds to the complementary distribution of the performance function at a given level γ. Each given γ uniquely determines α, and vice versa. A natural question in engineering is given a threshold γ on the performance, what is the probability of the system exceeding this threshold. Therefore, in the real computation, we pick γ first, and compute the corresponding risk $\alpha = P(\gamma)$. Note that γ should be less than $\mu_{\boldsymbol{\Delta}}(M)$, otherwise $P(\gamma) = 0$.

For a given performance level γ, a parameter choice Δ is considered as a "bad event" if $\overline{\lambda}_r(M\Delta) \geq \gamma$. Let $\boldsymbol{R_b}$ denote the "region of bad events", i.e.,

$$\boldsymbol{R_b} : = \{\Delta : \Delta \in \boldsymbol{B\Delta},\ \overline{\lambda}_r(M\Delta) \geq \gamma\}. \qquad (3.4)$$

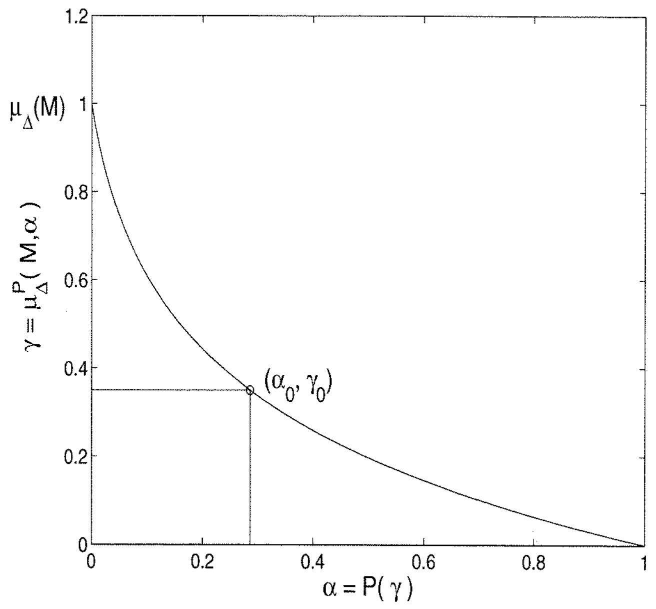

Figure 3.1: Definition of purely probabilistic μ.

Then

$$P(\gamma) = \int_{\boldsymbol{R_b}} p(\Delta) d\Delta, \tag{3.5}$$

where $p(\Delta)$ is the probability density function of Δ, so $\int_{\boldsymbol{B\Delta}} p(\Delta)d\Delta = 1$. For simplicity, we would assume that Δ is uniformly distributed in $\boldsymbol{B\Delta}$, in which case, $P(\gamma) = \frac{V(\boldsymbol{R_b})}{V(\boldsymbol{B\Delta})}$. Similarly, we can define the "region of good events" as

$$\boldsymbol{R_g} : = \{\Delta : \Delta \in \boldsymbol{B\Delta},\ \overline{\lambda}_r(M\Delta) < \gamma\}. \tag{3.6}$$

Since $V(\boldsymbol{R_g}) = V(\boldsymbol{B\Delta}) - V(\boldsymbol{R_b})$, all we need to find $P(\gamma)$ is to compute the volume of either $\boldsymbol{R_b}$ or $\boldsymbol{R_g}$. So the critical issue here is to characterize the separation of these two regions, i.e., the level surface of the function

$$\boldsymbol{L_\gamma} : = \{\Delta : \Delta \in \boldsymbol{B\Delta},\ \overline{\lambda}_r(M\Delta) = \gamma\}. \tag{3.7}$$

The definition of the purely probabilistic μ in [38] is slightly different, which is based on the invertibility of the matrix $I - \frac{M}{\gamma}\Delta$. Let

$$\boldsymbol{S}_\gamma: = \{\Delta : \Delta \in \boldsymbol{B\Delta},\ det(I - \frac{M}{\gamma}\Delta) = 0\} \tag{3.8}$$

be the boundary of singularity, dividing $\boldsymbol{B\Delta}$ into two subsets, then the volume of the subset that does not contain the origin defines probabilistic μ. The motivation behind this definition is that there is no real notion of stability in the purely probabilistic μ setting. The only issue here is the existence of nontrivial solutions in the loop equations from the feedback interconnection in Figure 2.2, which is determined by the singularity of $I - M\Delta$. There exist problems where there is a gap between the two surfaces $\boldsymbol{L}_\gamma$ and $\boldsymbol{S}_\gamma$ for γ sufficiently lower than $\mu_{\boldsymbol{\Delta}}(M)$. However, when γ is close to $\mu_{\boldsymbol{\Delta}}(M)$, $\boldsymbol{L}_\gamma$ and $\boldsymbol{S}_\gamma$ are always the same, which means the two separations are equivalent. Since we are mainly interested in the probability of rare events, we would assume that γ is close enough to the worst-case throughout this thesis, so we will not distinguish between $\boldsymbol{L}_\gamma$ and $\boldsymbol{S}_\gamma$ from now on, and refer to them as the "separating surface" between $\boldsymbol{R_b}$ and $\boldsymbol{R_g}$ with respect to certain performance level γ. This separation is illustrated in the following two-dimensional example.

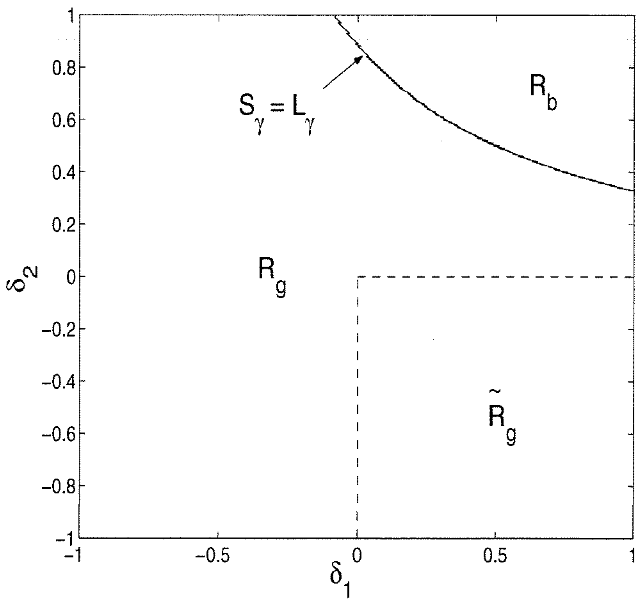

Figure 3.2: Separation of the parameter set.

Exact computation of $P(\gamma)$ depends on exact characterization of the above separation of $\boldsymbol{B\Delta}$, which is intractable in general. There are two alternative ways to approximate $P(\gamma)$. One is the Monte-Carlo simulation which samples the whole parameter set and estimates $P(\gamma)$ by the frequency of the occurrence of the "bad samples". This estimate will be referred to as the "soft bound", because it is neither an upper bound nor a lower bound, and it has no guarantees. When $P(\gamma)$ is extremely small, i.e., at the tail of the distribution, the Monte-Carlo method becomes highly inefficient because a large number of samples would be wasted on benign events.

Moreover, we may want some guaranteed information rather than just soft bounds, in which case hard bounds on the real distribution are highly desirable. The rest of this thesis is devoted to developing computable upper bounds on $P(\gamma)$. The basic idea is using the standard μ upper bound to identify regions where the matrix $I - M\Delta$ is guaranteed to be invertible, or the function $\overline{\lambda}_r(M\Delta)$ is guaranteed to be below γ, then this region has to be contained in the region of good events $\boldsymbol{R_g}$. One such region (separated by dashed lines) is demonstrated in Figure 3.2. Let $\tilde{\boldsymbol{R}}_{\boldsymbol{g}}$ denote the union of all such regions, and $\tilde{\boldsymbol{R}}_{\boldsymbol{b}} = \boldsymbol{B\Delta} \backslash \tilde{\boldsymbol{R}}_{\boldsymbol{g}}$. $\tilde{\boldsymbol{R}}_{\boldsymbol{g}}$ represents the guaranteed good region and $\tilde{\boldsymbol{R}}_{\boldsymbol{b}}$ represents the potentially bad region. Then $\tilde{\boldsymbol{R}}_{\boldsymbol{g}} \subseteq \boldsymbol{R_g} \Rightarrow \boldsymbol{R_b} \subseteq \tilde{\boldsymbol{R}}_{\boldsymbol{b}}$. Since $P(\gamma) = \frac{V(\boldsymbol{R_b})}{V(\boldsymbol{B\Delta})}$, $\frac{V(\tilde{\boldsymbol{R}}_{\boldsymbol{b}})}{V(\boldsymbol{B\Delta})}$ is an upper bound on $P(\gamma)$, which will be referred to as the "hard upper bound" in general. On the other hand, more accurate soft bounds can be obtained by only sampling the potentially bad region $\boldsymbol{R_b}$ instead of the whole parameter set. The improvement becomes more significant when the volume of $\tilde{\boldsymbol{R}}_{\boldsymbol{b}}$ becomes smaller. Both the hard upper bound and the soft bound can be refined using branch and bound type of algorithms, which is the topic of the next chapter.

A link exists between the above method and another variance reduction technique—importance sampling (IS). Instead of uniform sampling, an IS method chooses a proper distribution function $p(\cdot)$ that approximates the shape of the target function and generates samples based on this distribution. It is particularly useful for estimating the probability of rare events. In our method, we pick $p(\Delta) = 1$ for $\Delta \in \tilde{\boldsymbol{R}}_{\boldsymbol{b}}$, and 0 otherwise, which is a special case of general IS, also a stronger result since hard bound computation guarantees no information is lost when throwing away $\boldsymbol{B\Delta} \backslash \tilde{\boldsymbol{R}}_{\boldsymbol{b}}$.

Chapter 4 Branch and Bound

Branch and Bound (B&B) is a general technique for computing improved bounds for a given optimization problem. There is a history of applying B&B schemes for robustness analysis in control [7, 22, 67]. However these authors have not specifically addressed the problem of avoiding exponential growth in computational expense with problem size. In [54], Newlin and Young proposed using B&B to quickly improving the bounds for mixed μ with median-sized problems ($n < 100$), which aimed at producing a "practical" scheme with *typical* polynomial time computation. The effectiveness of this scheme was verified by extensive numerical experiments.

In general, B&B algorithms are useful for those optimization problems whose bounds depend on the domain of the problem. The idea is to refine the bounds on the global optimum by dividing the original domain of optimization into subdomains and computing local bounds on these new domains, until the gap between the global bounds is within certain tolerance. Experience has shown that the average quality of the bounds themselves is critical. The intuition behind this is that there are occasionally bad problems where the bounds are poor, but that branching creates new problems where the bounds are good. For this to be successful, the bounds must be good on average so that the branching process moves bad problems into easy problems. Interestingly, it has seemed less critical that the branching scheme be particularly clever. These points can be readily illustrated in the simple B&B algorithm described in the next section and the numerical experiment results shown afterwards.

4.1 A Naive Brand and Bound Algorithm

A naive branching scheme is shown in Figure 4.1. The algorithm consists of a simple heuristic to choose a branch variable, followed by splitting that variable into two equal

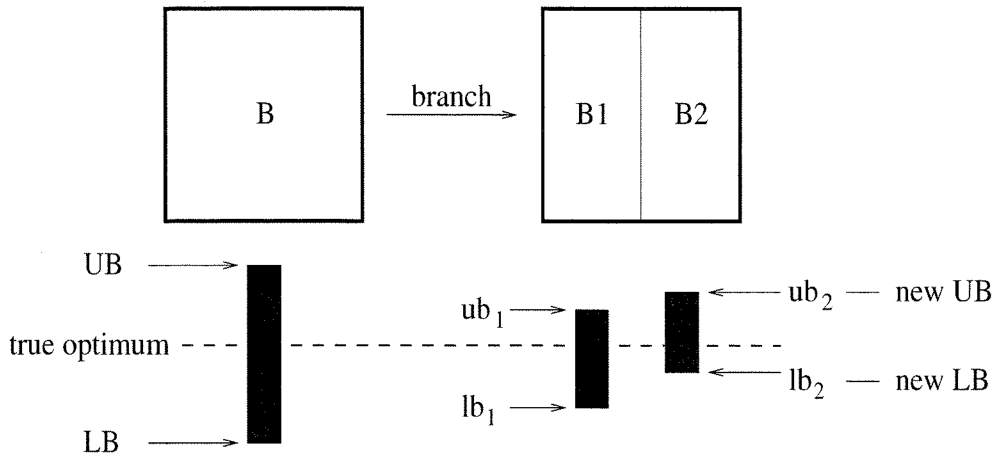

Figure 4.1: Branch and bound algorithm.

parts, creating two new independent problems on which the bounds are computed. (A more sophisticated algorithm would optimize both the variable chosen and the location of the cut.) The global lower bound (LB) is the maximum over all the local lower bounds (lb_i) and the global upper bound (UB) is the maximum over all the local upper bounds (ub_i). All the old and new branches form a tree structure, as shown in Figure 4.2. The branches that are crossed out on the tree are those that have a local upper bound below the global lower bound. This process is referred to as a "pruning process." It is essential that branches be pruned effectively to avoid exponential growth in the number of branches.

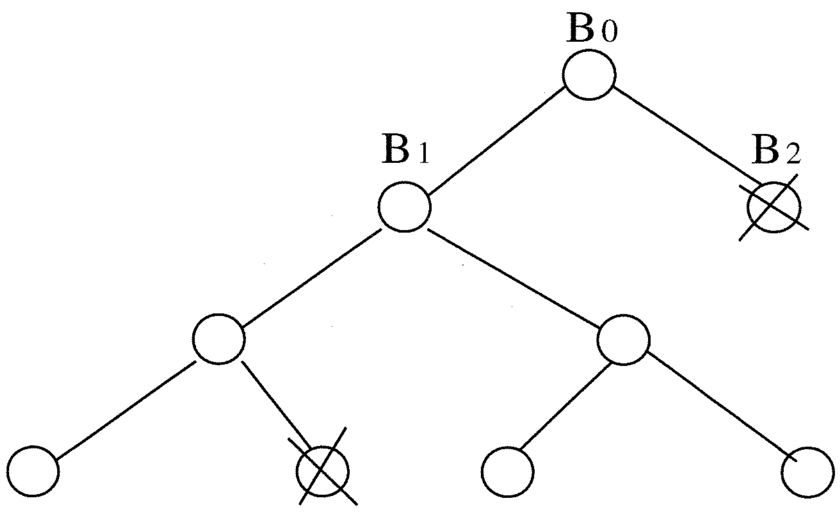

Figure 4.2: Tree of branches and the pruning process.

4.2 Branch and Bound for μ Computation

This section demonstrates how branch and bound can be used to effectively improve upper and lower bounds for μ in the worst-case framework. As our first numerical experiment, we computed these bounds on the purely real μ discussed in Section 2.2 for matrices M of size 4, 8, 16, 32, and 64, with 50 matrices of each size. The elements of the matrices were zero mean normally distributed pseudorandomly generated floating point numbers. The bounds either achieved a normalized error of 10^{-3} or a maximum number of iterations were run. Figure 4.3 shows the ratio of the lower bound (LB) to the upper bound (UB) for all the problems tested. The x-axis shows n — the number of real parameters in Δ, which is referred to as the size of the problem. Note that the bounds degrade somewhat with problem size. If the relative error in the bounds is defined as $e = 1 - \frac{LB}{UB}$, then for $n = 4$, most problems have $e < 10\%$, while for $n = 64$, all the problems have $e \geq 10\%$. However, even for $n = 64$, the error never exceeds 20%. This means the bounds have moderately good quality on average. Unfortunately, there are occasionally poor bounds for any size problem, where poor here means the ratio is not in the typical range for that size.

Figure 4.4 shows the results of applying the simple branch and bound scheme described in the previous section to the same problems as in Figure 4.3, with 10 branches performed on each problem. Note that this made substantial improvements in the quality of the bounds, with only a small fraction of the size 64 problems remaining with an error of greater than 10%. Figure 4.5 focuses on the worst problems for each problem size and plots the number of branches required to achieve a given level for the error, such as 15%, 10%, 5%, 2% and 1%. Since this is a log-log scale, straight lines indicate polynomial growth, and flat lines indicate no growth.

It is not clear what the growth rates are for small percentages, as the 2% and 1% cases where only done up to size 16. Beyond that, the computation time was too great (several hours per problem) to be practical on individual workstations. Note that the worst problem required exactly 3 branches for each problem size to reach 15%, and that 10% was easily achieved for all problem sizes. This further supports

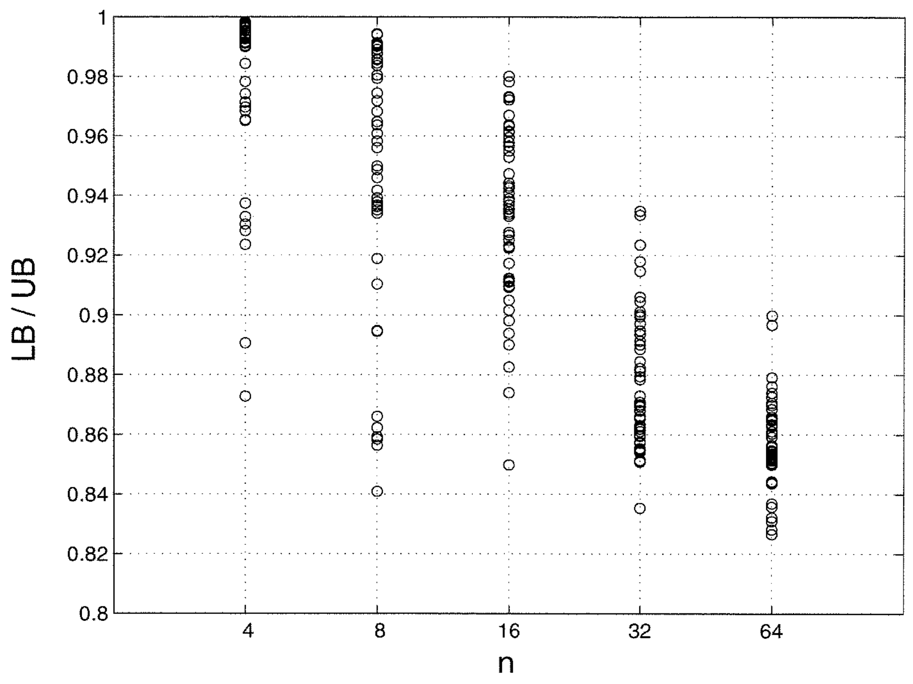

Figure 4.3: $\frac{LB}{UB}$ vs. problem size, with 50 problems of each size, without branching.

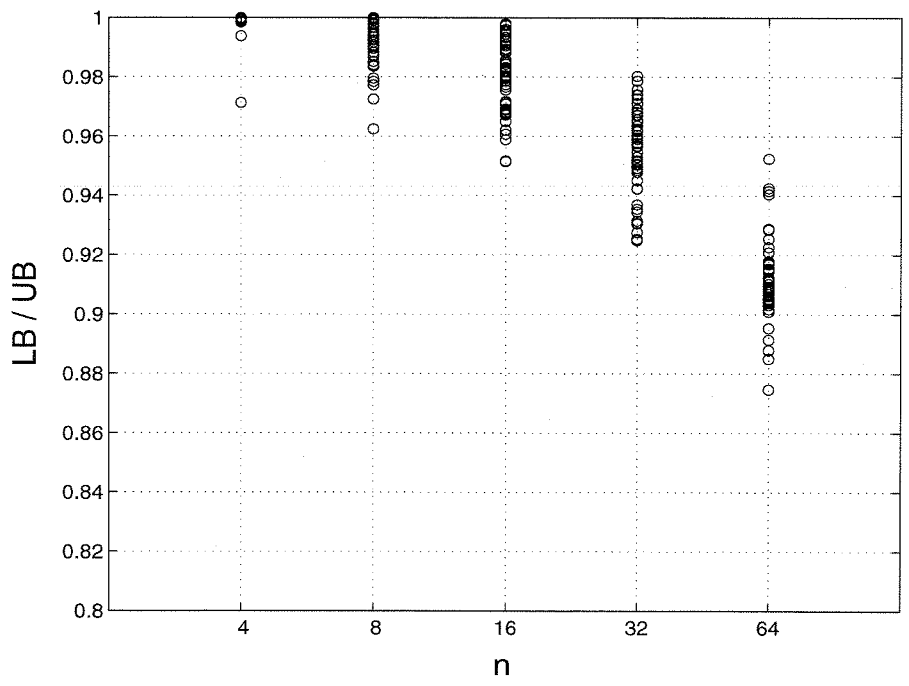

Figure 4.4: $\frac{LB}{UB}$ vs. problem size, for the same 50 problems of each size, after branching 10 times.

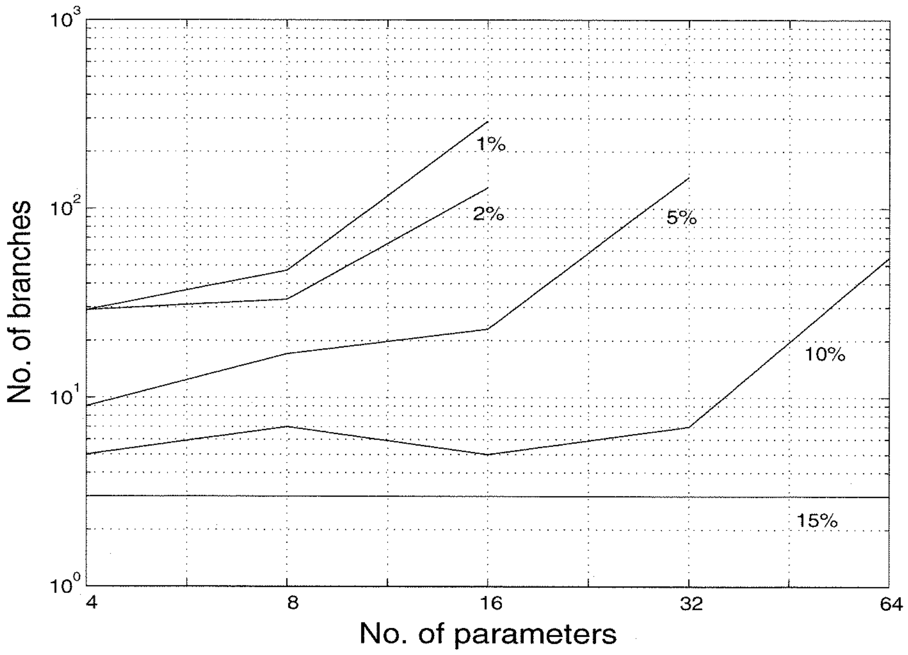

Figure 4.5: No. of branches vs. problem size for various tolerances, for the worst problem out of 50 in each size.

the notion that branch and bound can easily take the worst-case problems and get them to roughly the level that the bounds achieve on average, but not much better. It is possible that more sophisticated branching schemes would improve on this, and certainly the branch and bound algorithms are embarrassingly parallelizable, but we have not yet explored these possibilities. Of course, there must exist truly bad examples where even very clever branch and bound fails (or P=NP), but these seem so rare that they are very unlikely to be encountered in practice. This latter assertion must be supported by exactly the type of numerical experiments we are showing here.

The point of these numerical experiments is to underscore the point that branch and bound can be used effectively to overcome the inherent intractability of worst-case robustness analysis. The key seems to be to have good bounds, where good here means good on average. Even naive branching schemes can then be relatively effective in refining the bounds in those cases where they are poor.

4.3 Branch and Bound for Probabilistic μ — The BNB Algorithm

In this section the goal is to apply branch and bound techniques to probabilistic robustness analysis similar to the way it was used for worst-case. In Chapter 3, a function $P(\gamma)$ was defined as the complementary distribution of the function $\overline{\lambda}_r(M\Delta)$, which is the risk that certain performance threshold γ is exceeded. Using the idea proposed in the end of Chapter 3, two kinds of bounds can be computed on $P(\gamma)$ by using branch and bound, as illustrated in Figure 4.6.

Hard upper bound Suppose $\boldsymbol{B\Delta}$ is divided into two regions $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$. An upper bound on $P(\gamma)$ can be computed if the upper bound of $\overline{\lambda}_r(M\Delta)$ on one region, say, $\boldsymbol{B}_2$, is below γ. Then there is no contribution whatsoever from the region $\boldsymbol{B}_2$ to $P(\gamma)$. Hence, $\boldsymbol{R_b} \subseteq \boldsymbol{B}_1$, which leads to $P(\gamma) \leq \frac{V(\boldsymbol{B}_1)}{V(\boldsymbol{B\Delta})}$.

Soft bound A soft bound on $P(\gamma)$ can be obtained through the Monte-Carlo simulation in B_1, the potentially bad region. With the hard bound excluding benign regions of the parameter space, the soft bound can be produced more efficiently.

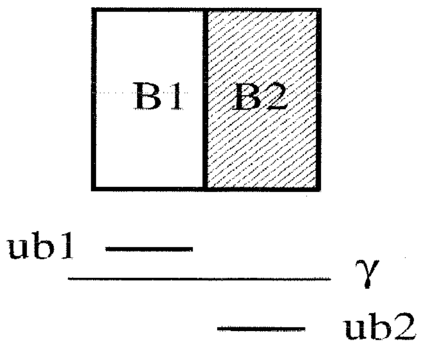

Figure 4.6: Branch and bound for probabilistic μ.

The following subsection describes how the above hard upper bound and the soft bound can be refined by a simple branch and bound algorithm. To distinguish it from other algorithms introduced in later chapters, it is referred to as "the BNB algorithm". The corresponding hard upper bound and soft bound are denoted by hub_{BNB} and sb_{BNB}, respectively.

4.3.1 The BNB Algorithm

1. The branching scheme is fairly similar to the one in the naive B&B algorithm in Section 4.1. This time, only standard μ upper bound is computed, and the branches that are pruned are those whose upper bounds are below γ.

2. Let $\tilde{\boldsymbol{R}}_{\boldsymbol{b}}$ be the union of the remaining branches, then $\boldsymbol{R}_{\boldsymbol{b}} \subseteq \tilde{\boldsymbol{R}}_{\boldsymbol{b}}$. Hence,

$$hub_{BNB} : = \frac{V(\tilde{\boldsymbol{R}}_{\boldsymbol{b}})}{V(\boldsymbol{B\Delta})} \tag{4.1}$$

3. Sample $\tilde{\boldsymbol{R}}_{\boldsymbol{b}}$ and compute $\overline{\lambda}_r(M\Delta)$ on each sample. Suppose the total number of samples is N and the number of samples for which $\overline{\lambda}_r(M\Delta) \geq \gamma$ is N_b. Then

$$sb_{BNB} : = \frac{N_b}{N} \times hub_{BNB}. \tag{4.2}$$

The soft bound can be used to test the tightness of the hard upper bound. On the other hand, as can be seen in (4.2), with the help of the hard bound, the effective sample size in the soft bound increases from N to $\frac{N}{hub_{BNB}}$. For extremely small $P(\gamma)$, if hub_{BNB} is a tight bound, the above improvement can be quite significant.

To illustrate the relationship between the hard upper bound and the soft bound, in Figure 4.7 the two bounds for a particular size 4 problem are shown. The matrix was normalized so that $\mu_{\boldsymbol{\Delta}}(M) = 1$. The y-axis indicates γ, and the x-axis (on a log_{10} scale) indicates the corresponding values of the hard upper bound and the soft bound on $P(\gamma)$. When γ varies from 0.9 to 1, two bounds on the tail of the true distribution are formed. In the figure the lower boundary of the rectangles represents the hard upper bound while the solid line represents the soft bound. As in the standard μ computation, there always exists a gap between the two kinds of bounds, which gives some measure of their quality. From the experience of a great number of numerical experiments we know that the extension of branch and bound techniques from the worst-case to probabilistic computation is not trivial. Some insights can be obtained in the two special cases discussed in the next subsection.

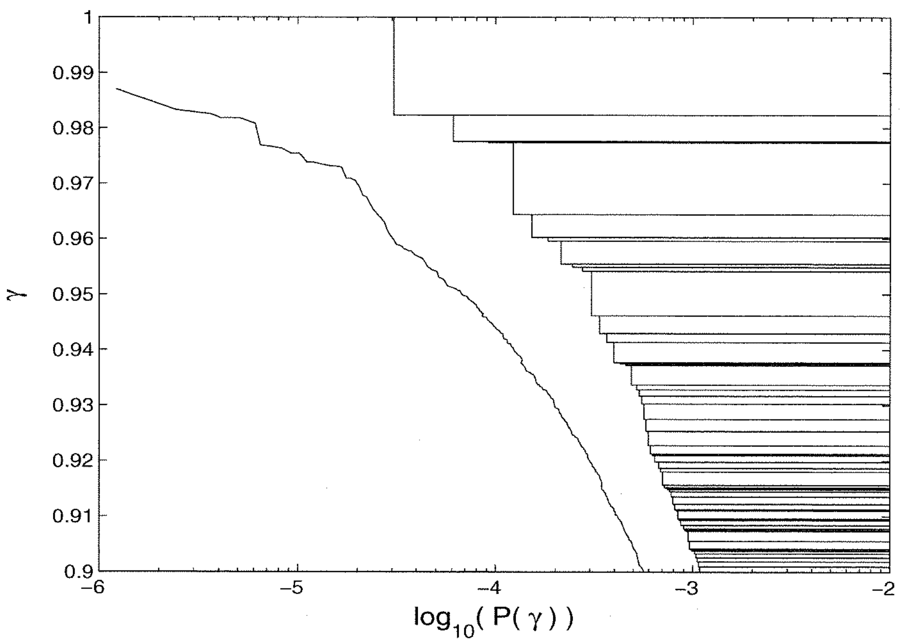

Figure 4.7: Hard upper bound and soft bound on $P(\gamma)$ for a size 4 problem, after branching 100 times, $0.9 \leq \gamma \leq 1$.

4.3.2 Two Special Cases

The two cases considered here are in a sense the extreme ends of the space of examples, but for which we can analytically compute probability distributions. Without loss of generality, assume $\mu_{\mathbf{\Delta}}(M) = 1$. Let $\epsilon = 1-\gamma$, then $P(\gamma) = \hat{P}(\epsilon) = P[\overline{\lambda}_r(M\Delta) \geq 1-\epsilon]$.

For $M = I_n$, $\overline{\lambda}_r(\Delta M) = \max(\delta_i)$. As an example the parameter set $\boldsymbol{B\Delta}$ for $n = 2$ is shown in Figure 4.8. The shaded area indicates the region where $\overline{\lambda}_r(M\Delta) \geq 1 - \epsilon$. It is easy to see that

$$\hat{P}(\epsilon) = 1 - (1 - \frac{\epsilon}{2})^n. \tag{4.3}$$

Note that $\hat{P}(\epsilon) \to 1$ as $n \to \infty$, which means the bad events become dominant in terms of probability for high dimensional problems. In standard μ computation the bounds are exact so there is no need to branch. It is hard to use the naive branch and bound algorithm to get hard bounds on $\hat{P}(\epsilon)$ though, as the function achieves its maximum on all the faces of $\boldsymbol{B\Delta}$ with any $\delta_i = 1$. Whichever parameter we choose to cut, both of the two new branches have the same bound.

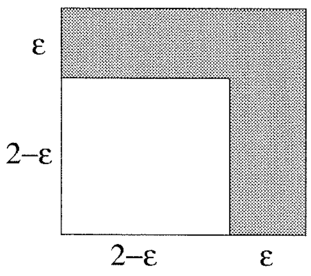

Figure 4.8: Special case 1 — M is identity.

As the opposite extreme suppose M is rank-one and has already been scaled by DMD^{-1} so that $M = aa^T$, where $a = [a_1 \; a_2 \; \cdots \; a_n]^T \in \mathbb{R}^n$ and $\|a\| = 1$. Then

$$\bar{\lambda}_r(\Delta M) = \sum_{i=1}^{n} a_i^2 \delta_i. \tag{4.4}$$

Note that this function is linear in the parameters δ_i. The importance of this fact will be discussed later. The worst case is achieved at the vertex where $\delta_i = 1$, for all i. Again, the bounds for the standard μ are exact so no branching is required. It is easily checked, however, that our branch and bound scheme to bound the probability distribution is prohibitively expensive, even though we can compute $\hat{P}(\epsilon)$ analytically. In the simplest case where $a_i^2 = \frac{1}{n}$, for all i, it can be shown that

$$\hat{P}(\epsilon) \le \frac{(\frac{n\epsilon}{2})^n}{n!}, \tag{4.5}$$

where the equality holds for $\epsilon \le \frac{2}{n}$. It is possible to get $\hat{P}(\epsilon)$ in general but the formula is messy. It is easy to show that this is the hardest case ($\hat{P}(\epsilon)$ is smallest for fixed ϵ and n) for all the rank-one matrices with $\mu = 1$. For general rank-one problems we have

$$\hat{P}(\epsilon) = \frac{(\frac{\epsilon}{2})^n / \prod_{i=1}^{n} a_i^2}{n!} \ge \frac{(\frac{n\epsilon}{2})^n}{n!}, \tag{4.6}$$

if $\frac{\epsilon}{2a_i^2} \le 1, \; i = 1, 2, \ldots, n$. The minimum is achieved when $a_i^2 = \frac{1}{n}$, for all i, which is exactly the simplest case considered above. Note that in contrast to the M being identity case, $\hat{P}(\epsilon) \to 0$ very rapidly as $n \to \infty$, which means the bad events become extremely rare as the size of the problem grows bigger.

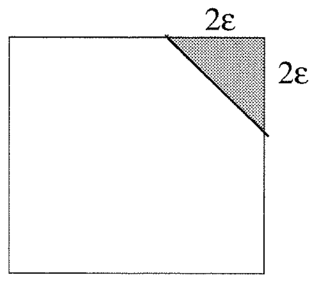

Figure 4.9: Special case 2 — M is rank-one.

In both cases, it is trivial to obtain bounds for μ in the worst-case formulation, and apparently extremely difficult to compute hard bounds on probabilistic μ by the naive application of branch and bound. The rank-one problem appears particularly problematic, and we would expect general random matrices to have similar characteristics. As the first step in verifying this, we generated 20 random matrices of size 4 with 10 of them being rank-one and 10 of them not. All of them were normalized to have $\mu_{\mathbf{\Delta}}(M) = 1$. Then we computed the soft bound on $P(\gamma)$ (or $\hat{P}(\epsilon)$) for $0.95 < \gamma < 1$ in the neighborhood of the worst case. The average values of the soft bounds are shown in Figure 4.10, with the dashed line and the solid line representing the rank-one matrices and the general random matrices, respectively. The dotted line represents the theoretical value of $P(\gamma)$ for the worst rank-one problem as in (4.5). This plot clearly shows that in the neighborhood of the worst-case, the general matrices have very similar characteristics as the rank-one matrices. Unfortunately, we know form the previous analysis that we cannot easily compute accurate hard bounds on the rank-one problem, so we can reasonably anticipate having similar difficulties with general random matrices. This has been verified by numerical experiments, as is shown in the next subsection.

4.3.3 Numerical Experiments

The BNB algorithm was tested on 80 random matrices of size 2, 4, 6, and 8, with 20 matrices of each size. The matrices were generated using MATLAB 'randn' command. The results are shown in Figure 4.11 and Figure 4.12. We set $\gamma = \eta * \mu_{\mathbf{\Delta}}(M)$ and

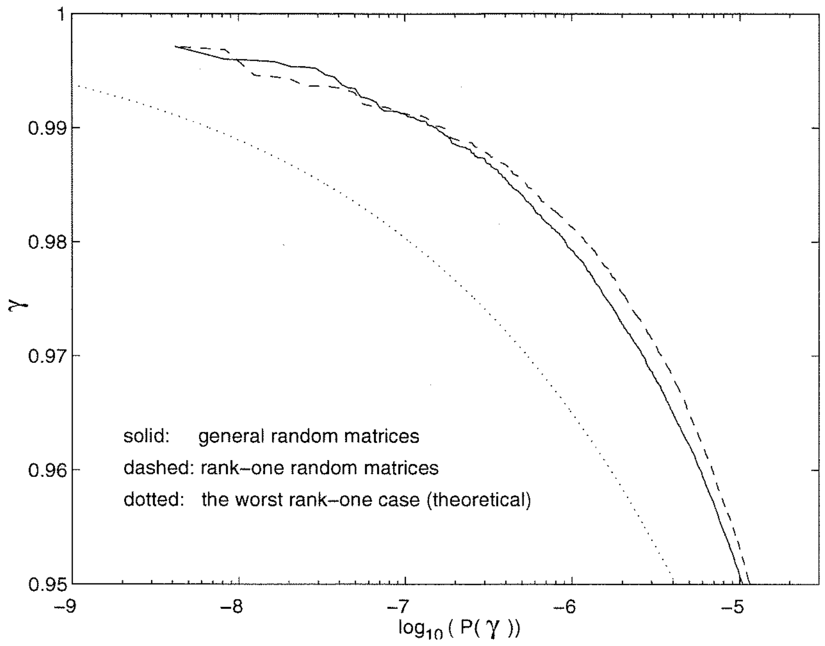

Figure 4.10: Soft bounds on $P(\gamma)$ for different kinds of problems for $n = 4$.

$\eta = 0.9$ so that γ is close to the worst case. When $\mu_{\mathbf{\Delta}}(M)$ is not available, the lower bound for $\mu_{\mathbf{\Delta}}(M)$ is used. For each problem $\boldsymbol{B\Delta}$ is branched 200 times. The ratio of sb_{BNB} to hub_{BNB} is denoted by rt_{BNB}. This notation is used for the results from the other methods in later chapters as well.

In Figure 4.11, instead of comparing the bounds for each individual problem, we focus more on the overall performance of the algorithm by showing all the bounds together. Each mark ('x' or 'o') corresponds to one problem. The hub_{BNB} is shifted slightly to the left for easiness of comparison. Obviously the sb_{BNB} decreases exponentially with the size of the problem n, while the average hub_{BNB} hardly varies with n. The gap between them becomes larger very quickly as n increases. For $n = 6$ and $n = 8$, hub_{BNB} and sb_{BNB} are orders of magnitude apart. Note that for $n = 8$ only 18 data points are displayed, since $sb_{BNB} = 0$ for the other two problems, meaning none of the samples hit $\boldsymbol{R_b}$. Figure 4.12 shows the ratio between these two bounds for all the problems, with each 'o' representing one problem. The rt_{BNB} decays exponentially with n, with an average of 0.8001, 0.1986, 0.0090 and 0.0004 for size 2, 4, 6 and 8 respectively, indicating the fast performance degradation with the growth of

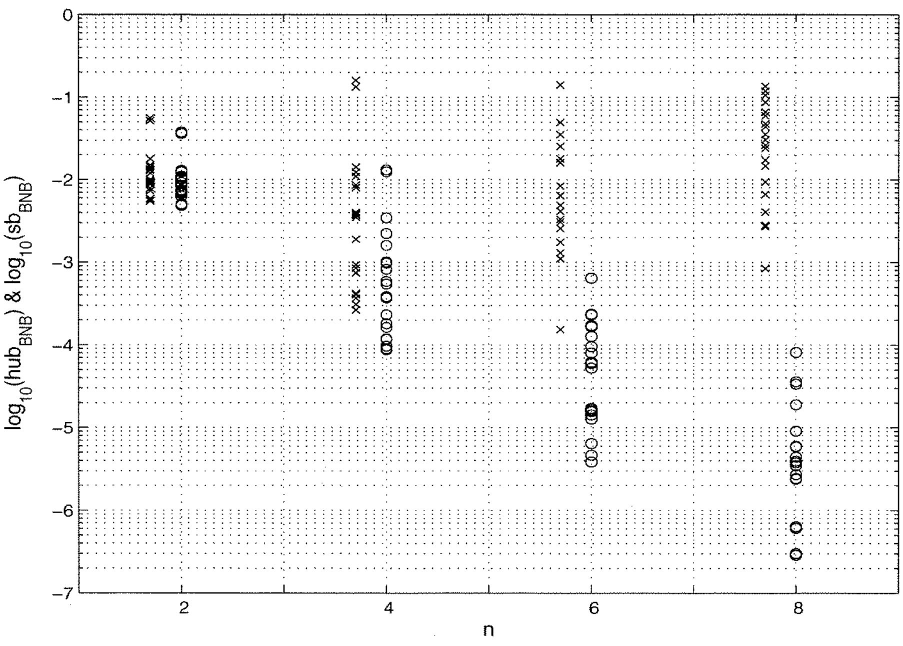

Figure 4.11: hub_{BNB}('x') and sb_{BNB}('o') vs. problem size, with 20 problems of each size.

the problem size. For $n = 8$ this average was computed using 18 problems because the other two have $rt_{BNB} = 0$.

A natural suspicion for the poor performance of the BNB algorithm is that the quality of the bounds for the worst-case μ are not accurate enough to exclude the benign regions effectively. But further numerical experiments suggest that this is not the case. First, we can get fairly tight (within 1%) worst-case bounds after branching a certain number of times for problems of the size we were testing. Second, recall that we can compute exact bounds by checking all the vertices of $\boldsymbol{B\Delta}$, which is feasible for $n \leq 8$. Thus we can recompute the probability bounds on the same matrices used above but with exact worst-case bounds and compare the results. We did this and it made no significant difference.

The question is: What is the intrinsic difficulty behind the apparent intractability of the computation? Interestingly, this difficulty seems to be present even in rank-one problems, which are both trivial from a worst-case perspective and can be treated analytically. According to the previous analysis, in the rank-one case, the performance function is linear in the uncertain parameters, which means that the surface $\boldsymbol{S}_\gamma$ that

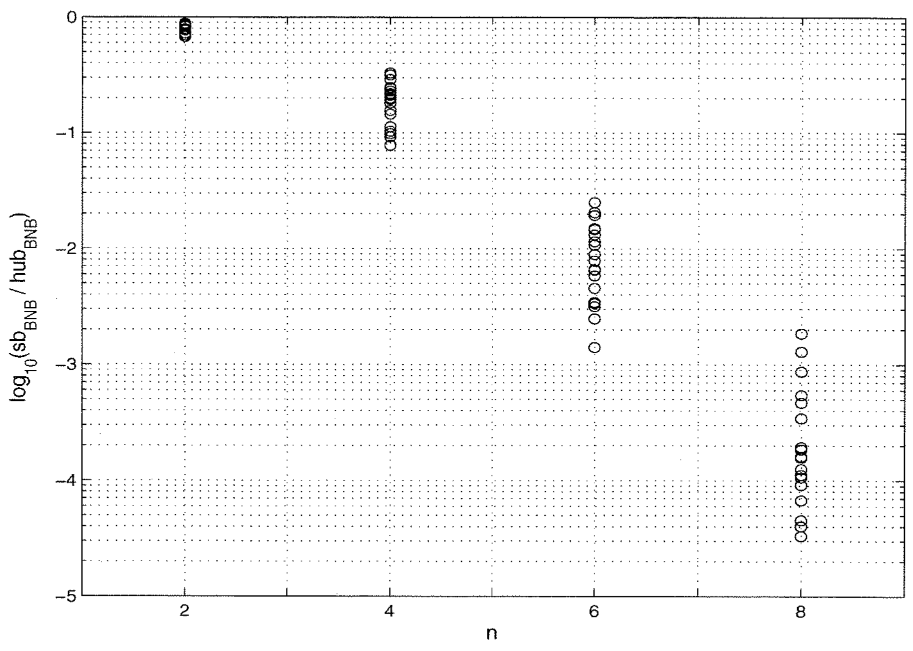

Figure 4.12: $rt_{BNB} = \frac{sb_{BNB}}{hub_{BNB}}$ vs. problem size, for the same 20 problems of each size.

separates $\boldsymbol{R_b}$ from $\boldsymbol{R_g}$ is a hyperplane in $\mathbb{R}^n$. An example is shown in Figure 4.13. The true region of bad events $\boldsymbol{R_b}$ is a simplex at the corner of $\boldsymbol{B\Delta}$. The shaded region is $\tilde{\boldsymbol{R}}_{\boldsymbol{b}}$, a union of hypercubes used in the BNB algorithm to capture $\boldsymbol{R_b}$. However, after branching 100 times, there is still quite a gap between these two sets. Essentially what the BNB algorithm does is gridding the separating surface $\boldsymbol{S_\gamma}$ using axially aligned cuts, which is conceivably not an intelligent way to approximate a hyperplane. Instead, a single linear cut along the hyperplane will isolate $\boldsymbol{R_b}$ immediately. For this separation to be feasible, we need to develop a way to bound the performance function on a more exotic region than just a hypercube. However, the standard μ framework only concerns uncertainty set with bounded ∞-norm. This is the motivation for introducing μ with linear cuts in the next chapter.

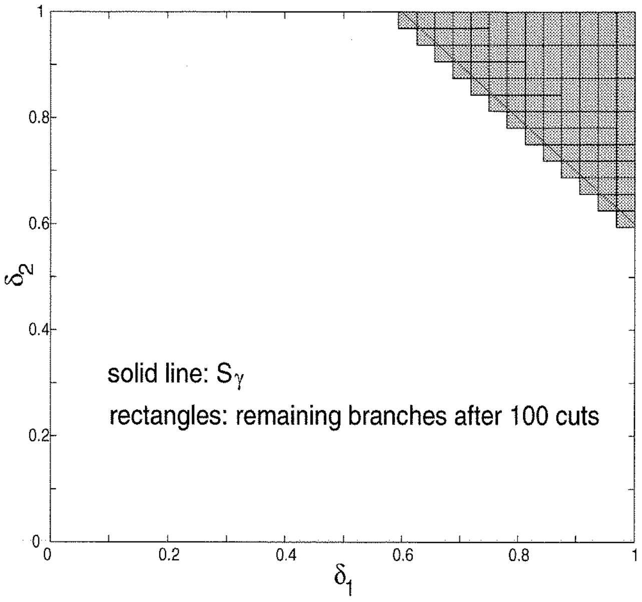

Figure 4.13: A rank-one example, $\mu_{\mathbf{\Delta}}(M) = 1$ and $\gamma = 0.8$.

Chapter 5 μ with Linear Cuts

The need to compute hard bounds on purely probabilistic μ for rank-one problems motivated the research of μ with uncertainties in more exotic regions than the standard ∞-norm bounded sets, for example, spherical μ [39], elliptical μ [39] and μ with linear cuts [82]. Among these extensions to the standard μ framework, μ with linear cuts is the most relevant to this thesis, which considers real parametric uncertainty with linear constraints. Its definition is described in the first section. Then three methods are presented to develop computable upper bounds for μ with linear cuts. Comparison of these three methods will be shown through numerical examples in the last section.

5.1 Definition

Let $\delta = [\delta_1, \ldots, \delta_n]^T \in \mathbb{R}^n$ be the diagonal of the real parametric uncertainty Δ. In addition to the standard ∞-norm constraint on Δ, we consider the following linear constraint in the space of δ:

$$|c^T \delta| \leq 1, \tag{5.1}$$

where $c = [c_1, c_2, \ldots, c_n]^T \in \mathbb{R}^n$ is the constant coefficient vector with $||c|| = 1$. The quantity $|c^T\delta|$ is a semi-norm on δ. For any constant $\beta \in \mathbb{R}$, the set $\{\delta : c^T\delta = \beta\}$ defines a hyperplane in $\mathbb{R}^n$ to which the distance from the origin is $|\beta|$. The vector c is the normal vector of this hyperplane. Therefore, (5.1) defines the closed region in $\mathbb{R}^n$ between the two hyperplanes $\{\delta : c^T\delta = \pm 1\}$.

Let $\boldsymbol{B\Delta}_{lc}$ denote the set defined by a combination of the standard ∞-norm constraint and the linear constraint in equation (5.1), i.e.,

$$\boldsymbol{B\Delta}_{lc}: = \{\Delta \in \boldsymbol{\Delta} : \|\Delta\| \leq 1 \ \& \ |c^T\delta| \leq 1\}. \tag{5.2}$$

Definition 5.1 *For $M \in \mathbb{R}^{n\times n}$, μ with linear cuts is defined as*

$$\mu_{\boldsymbol{\Delta},lc}(M): = \frac{1}{\min\{\beta : \Delta \in \boldsymbol{B\Delta}_{lc},\ \det(I - \beta M\Delta) = 0\}} \tag{5.3}$$

unless $det(I - M\Delta) \neq 0, \forall \Delta \in \boldsymbol{\Delta}$, in which case $\mu_{\boldsymbol{\Delta},lc}(M): = 0$.

In the other words, the interconnected system (S1) is well-posed for all $\Delta \in \frac{1}{\beta}\boldsymbol{B\Delta}_{lc}$ if and only if $\mu_{\boldsymbol{\Delta},lc}(M) < \beta$, where $\frac{1}{\beta}\boldsymbol{B\Delta}_{lc}: = \{\Delta \in \boldsymbol{\Delta} : \|\Delta\| \leq \frac{1}{\beta}$ and $|c^T\delta| \leq \frac{1}{\beta}\}$. If only the linear constraint $|c^T\delta| \leq \frac{1}{\beta}$ is scaled while the unit box $\boldsymbol{B\Delta}$ remains fixed, the skew version of $\boldsymbol{B\Delta}_{lc}$ can be defined as

$$\boldsymbol{B\Delta}^s_{lc,\beta}: = \{\Delta \in \boldsymbol{\Delta} : \|\Delta\| \leq 1 \text{ and } |c^T\delta| \leq \frac{1}{\beta}\}. \tag{5.4}$$

Definition 5.2 *For $M \in \mathbb{R}^{n\times n}$, the skew version of μ with linear cuts is defined as*

$$\mu^s_{\boldsymbol{\Delta},lc}(M): = \frac{1}{\min\{|c^T\delta| : \Delta \in \boldsymbol{B\Delta}, \det(I - M\Delta) = 0\}} \tag{5.5}$$

unless $det(I - M\Delta) \neq 0, \forall \Delta \in \boldsymbol{B\Delta}$, in which case $\mu^s_{\boldsymbol{\Delta},lc}(M): = 0$.

Equivalently, the interconnected system (S1) is well-posed for all $\Delta \in \boldsymbol{B\Delta}^s_{lc,\beta}$ if and only if $\mu^s_{\boldsymbol{\Delta},lc}(M) < \beta$.

Like in the standard μ case, computing either $\mu_{\boldsymbol{\Delta},lc}$ or $\mu^s_{\boldsymbol{\Delta},lc}$ is NP-hard. Since it is more convenient to use the skew version of μ with linear cuts when applying branch and bound in probabilistic μ bound computation, we will focus in the rest of this chapter on obtaining a computable upper bound on $\mu^s_{\boldsymbol{\Delta},lc}$. Three methods for achieving this goal will be described in the next section. All of them can be easily modified to provide an upper bound on $\mu_{\boldsymbol{\Delta},lc}$ as well.

5.2 Upper Bound Computation

5.2.1 Elliptical Cut

The linear constraint defined in (5.1) can be viewed as the limit of a sequence of hyperellipsoids in the parameter space as the eccentricity goes to infinity. So the recent result of spherical μ presented in [39] can be used to obtain an upper bound on μ with linear cuts. Spherical μ is a generalization of the standard μ, with spherical constraints on the uncertainty. Although it is a special case of the generalized μ defined in [16], an LMI-based upper bound for the spherical μ with a general "nominal system" M was provided in [39], while [16] only discussed the case when M is a rank-one matrix. The above result was also generalized to μ with ellipsoidal constraints

$$\delta^T P \delta \leq 1, P > 0. \tag{5.6}$$

Definition 5.3 *For $M \in \mathbb{R}^{n \times n}$, μ for a hyperellipsoid is defined by*

$$\mu_{\mathbf{\Delta},e}(M) \colon = \frac{1}{\min\{\sqrt{\delta^T P \delta} : \Delta \in \mathbf{\Delta}, \det(I - M\Delta) = 0\}}. \tag{5.7}$$

An upper bound on $\mu_{\mathbf{\Delta},e}(M)$ is given by

$$\overline{\mu}_{\mathbf{\Delta},e}(M) = \inf_{D>0}\{\beta \colon M^T(P^{-1} \circ D)M < \beta^2 D\}. \tag{5.8}$$

The construction of using elliptical cuts to approximate linear cuts follows. Let $T = [c\ C_\perp]^T$, where c is as defined in (5.1), and $C_\perp$ is the matrix whose columns form an orthonormal basis for $Ker(c^T)$. So T is a unitary matrix. Let

$$P = T^{-1}\Sigma T = T^{-1} * diag[1, \frac{1}{\sigma_2^2}, \ldots, \frac{1}{\sigma_n^2}] * T. \tag{5.9}$$

Then the level set $\boldsymbol{L}_\beta = \{\delta : \sqrt{\delta^T P \delta} = \frac{1}{\beta}\}$ describes the hyperellipsoid with $\frac{1}{\beta}, \frac{\sigma_2}{\beta}, \ldots, \frac{\sigma_n}{\beta}$ being the lengths of the axes. When $\sigma_i \to \infty (i = 2, \ldots, n)$, $\boldsymbol{L}_\beta$ approaches the hyperplanes $\{\delta : c^T\delta = \pm\frac{1}{\beta}\}$. So the bound in (5.8) with P in the extreme case provides an upper bound on μ with purely linear constraints. By combining this bound with the standard μ upper bound in (2.15), an upper bound on $\mu^s_{\mathbf{\Delta},lc}(M)$ can be constructed as

$$\overline{\mu}^s_{\mathbf{\Delta}}, lc(M) = \inf_{D_1 \in \boldsymbol{D}, D_2 > 0} \{\beta > 0 \colon M^T(D_1 + P^{-1} \circ D_2)M < D_1 + \beta^2 D_2\}. \tag{5.10}$$

However, with the high eccentricity of the ellipsoid, the upper bound achieved is very conservative. The conservativeness decreases when the eccentricity becomes lower, at the cost of worse approximation of the hyperplanes. The method implemented intersects a high eccentricity ellipse with a less eccentric one to try to improve the performance, which results in a nonconvex optimization problem because of the freedom in the choice of the ellipses.

5.2.2 Implicit Method

For μ with linear cuts, (5.1) represents a constraint in the operator space Δ, which must be converted to a signal constraint to cast $\mu^s_{\mathbf{\Delta},lc}$ as a standard implicit μ problem. Based on this intuition, the implicit system in Figure 5.1 can be constructed to perform linear cuts on $\boldsymbol{B\Delta}$.

Note that $k^T \in \mathbb{R}^n$ is an arbitrary row vector, and $K = [1, 1, \ldots, 1]^T * k^T \in \mathbb{R}^{n\times n}$.

Theorem 5.4

$$\mu_{\tilde{\mathbf{\Delta}}}(\tilde{C}, \tilde{M}) < \beta \implies \mu_{\mathbf{\Delta},lc}(M) < \beta$$

where $\tilde{\mathbf{\Delta}} = \{\begin{bmatrix} \Delta & & \\ & \Delta & \\ & & \delta_0 \end{bmatrix} : \Delta \in \mathbf{\Delta}, \delta_0 \in \mathbb{R}\}$, $\tilde{M} = \begin{bmatrix} M & 0 & 0 \\ K & 0 & 0 \\ k^T & 0 & 0 \end{bmatrix}$, $\tilde{C} = [0 \; c^T \; 1]$.

Proof:

First, it is easy to show that for any $\Delta \in \frac{1}{\beta}\boldsymbol{B\Delta}_{lc}$, there exists a δ_0 such that $\tilde{\Delta} \in \frac{1}{\beta}\boldsymbol{B}\tilde{\mathbf{\Delta}}$ by letting $\delta_0 = c^T\delta$, hence, $\bar{\sigma}(\Delta) \le \frac{1}{\beta}$ and $|c^T\delta| \le \frac{1}{\beta} \implies \bar{\sigma}(\tilde{\Delta}) \le \frac{1}{\beta}$.

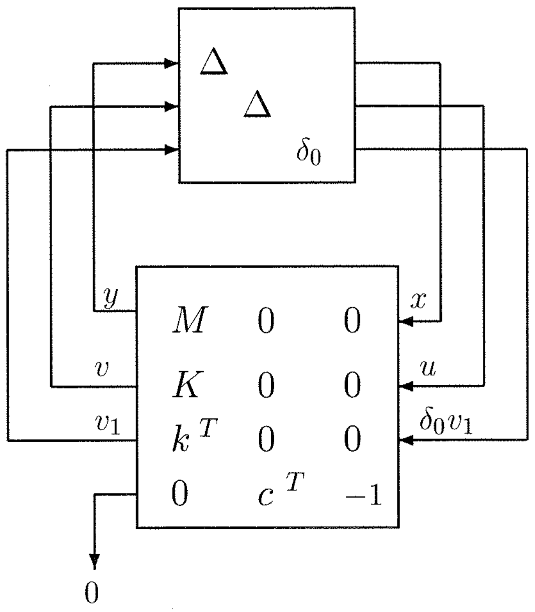

Figure 5.1: Implicit system (S3) constructed for linear cuts.

Second, we claim that for any $\tilde{\Delta} \in \frac{1}{\beta}\boldsymbol{B}\tilde{\boldsymbol{\Delta}}$ and any $x \in \mathbb{R}^n \backslash \{0\}$, let $v_1 = k^T x$, then

$$(c^T\delta - \delta_0)v_1 = 0 \ \Rightarrow \ (I - \Delta M)x \neq 0. \tag{5.11}$$

Now we show this claim is true.

$$\mu_{\tilde{\boldsymbol{\Delta}}}(\tilde{C}, \tilde{M}) < \beta$$

$$\Longrightarrow \qquad \forall \tilde{\Delta} \in \frac{1}{\beta}\boldsymbol{B}\tilde{\boldsymbol{\Delta}},\ Ker \begin{bmatrix} I - \tilde{\Delta}\tilde{M} \\ \tilde{C} \end{bmatrix} = 0$$

$$\Longrightarrow \ \forall z \in \mathbb{R}^{(2n+1)} \backslash \{0\},\ \tilde{C}z = 0 \ \Rightarrow \ (I - \tilde{\Delta}\tilde{M})z \neq 0 \tag{5.12}$$

With the interconnection in (S3), for any $x \in \mathbb{R}^n \backslash \{0\}$, we can define

$$z = \begin{bmatrix} x \\ u \\ \delta_0 v_1 \end{bmatrix},$$

where $v_1 = k^T x \in \mathbb{R}$ and $u = \Delta v = \delta v_1 \in \mathbb{R}^n$.

Simple calculation shows that

$$\begin{aligned} \tilde{C} z = 0 \quad &\Longleftrightarrow \quad (c^T \delta - \delta_0) v_1 = 0, \\ (I - \tilde{\Delta}\tilde{M}) z = 0 \quad &\Longleftrightarrow \quad (I - \Delta M) x = 0. \end{aligned}$$

Hence, (5.12) holds implies that (5.11) holds.

Combining the above two steps together, we have shown that for any $\Delta \in \frac{1}{\beta}\boldsymbol{B\Delta}_{lc}$ and any $x \in \mathbb{R}^n \backslash \{0\}$, $(I - \Delta M)x \neq 0$, i.e., $det(I - \Delta M) \neq 0$. Hence, $\mu_{\boldsymbol{\Delta},lc}(M) < \beta$.
□

Remark: The converse of the above theorem is not necessarily true. It depends on the free parameter k in the implicit system. Specific conditions on k can be imposed to make the converse true, which will not be discussed here since it is not critical to our goal — finding a computable upper bound for μ with linear cuts. This theorem tells us that an upper bound for the implicit system (S3) is also an upper bound on $\mu_{\boldsymbol{\Delta},lc}(M)$. To compute an upper bound for the $\mu^s_{\boldsymbol{\Delta},lc}(M)$, the skew version of implicit μ is used, whose upper bound is identical to the one in (2.18) with β^2 replaced by $\begin{bmatrix} I_{2n} & \\ & \beta^2 \end{bmatrix}$.

The computation of $\bar{\mu}^s_{\boldsymbol{\Delta},lc}(M)$ involves the following issues:

- Permutations on the rows and columns of $\tilde{M}$ are needed such that $\tilde{\Delta}$ consists of n 2×2 repeated real scalar blocks and one real scalar. The upper bound formula actually used in the computation is a bit more involved than (2.18) due to the existence of the repeated scalar blocks. Proper generalization involves the use of both D and G scaling matrices, in a way that is similar to the mixed

μ upper bound in (2.11).

- The quality of the bounds depends largely on the choice of the vector k in $\tilde{M}$. If k is included into the optimization, the problem becomes biconvex, i.e., it is convex in D and G for fixed k, and convex for k for fixed D and G. There are no guarantees in finding the global optimum. A good heuristic is to pick k as the input vector of M corresponding to its maximum singular value to make $\tilde{M}$ more like a rank-one matrix, due to the fact that the upper bound is exact when $\tilde{M}$ is rank-one. And the performance of the upper bounds achieved is much better than those with random chosen k. When the bound achieved with the above k is still not good enough, the algorithm optimizes over D, G and k alternatively to improve the bound.

- It may be useful to have redundant constraints to improve the computation.

5.2.3 Parallelogram Method

The third method is fairly similar to the implicit formulation, in the sense that both are converting the linear constraint $|c^T\delta| \le 1$ into the norm constraint on a scalar δ_0, and the matrix dimension is augmented in both cases. The difference is, in this method, only the standard μ computation is involved. The cost is some extra regions outside $\boldsymbol{B\Delta}_{lc}$ have to be included when checking for singularity of $I - M\Delta$.

Again define

$$\delta_0 = c_1\delta_1 + c_2\delta_2 + \cdots + c_n\delta_n. \tag{5.13}$$

Therefore,

$$|c^T\delta| \le 1 \Longleftrightarrow |\delta_0| \le 1.$$

Let $\hat{\Delta} = diag[\delta_0, \delta_2, \delta_2, \ldots, \delta_n, \delta_n]$, then

$$\Delta = P_L\hat{\Delta}P_R, \tag{5.14}$$

where

$$P_L = \begin{bmatrix} 1 & -c_2 & 0 & -c_3 & 0 & \cdots & -c_n & 0 \\ 0 & 0 & 1 & & & & & \\ & & & 0 & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 0 & 1 \end{bmatrix}, P_R = \begin{bmatrix} \frac{1}{c_1} & \frac{1}{c_1} & 0 & \frac{1}{c_1} & 0 & \cdots & \frac{1}{c_1} & 0 \\ 0 & 0 & 1 & & & & & \\ & & & 0 & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 0 & 1 \end{bmatrix}^T .$$

Let $\hat{M} = P_R M P_L$. Then the skew μ upper bound on $\mu_{\hat{\Delta}}(\hat{M})$ is also an upper bound on $\mu^s_{\Delta,lc}(M)$. This follows from the definition of μ and the existence of non-trivial solutions to the loop equations. Be noted that the unit ball $\boldsymbol{B}\hat{\boldsymbol{\Delta}}$ does not map to $\boldsymbol{B\Delta}_{lc}$ using (5.14). The mapping of $\boldsymbol{B}\hat{\boldsymbol{\Delta}}$ contains $\boldsymbol{B\Delta}_{lc}$. A simple example for $n = 2$ is shown in Figure 5.2. The potential conservativeness of this bound is obvious.

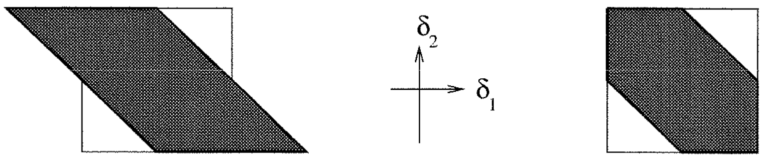

Figure 5.2: $\boldsymbol{B}\hat{\boldsymbol{\Delta}}$ and $\boldsymbol{B\Delta}_{lc}$.

In this example the norm constraint on δ_1 is lost when the problem is reformulated. Leaving out different δ_i's will result in different extra regions. If the δ_i with the maximum $|c_i|$ is chosen, the volume of the extra region is minimized, which will be a heuristic used to, hopefully, minimize the conservativeness of the bound. This heuristic is supported by numerical experience. The excess regions can be excluded by adding an implicit constraint as presented in Section 5.2.2.

5.3 Numerical Examples

Rank-one problems are the motivation for doing linear cuts. So random rank-one matrices were used to test the effectiveness of the above methods. It turns out, like the standard μ upper bound, the upper bounds achieved on $\mu^s_{\Delta,lc}(M)$ with all the three methods are exact when M is rank-one, which is not surprising because essentially the elliptical cut and implicit methods are just extensions of the standard μ upper bound, and the parallelogram method employs the standard μ computation directly, and the conservativeness caused by extra regions does not exist in the rank-one case, assuming the cuts are appropriately aligned with the level sets.

For general random matrices, where each element is random, the relative performance of the different methods varies with the problem. Figure 5.3 shows the results for a particular random problem of size 2, for which the elliptical cut method works better than the implicit method, while the parallelogram method does not achieve any bound because of its conservativeness. In another size 2 example shown in Figure 5.4, both the implicit method and the parallelogram method achieve the exact bound while the performance of the elliptical cut is fairly poor.

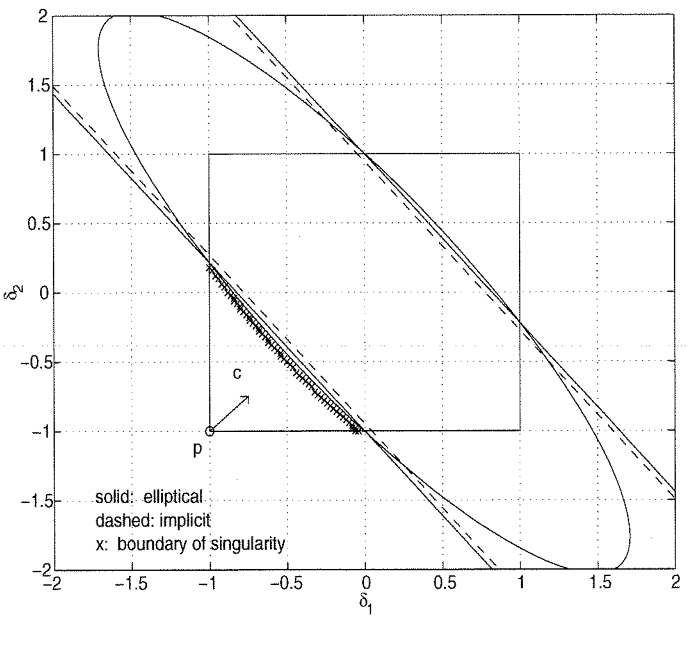

Figure 5.3: Upper bound for μ with linear cuts: Example 1.

For both problems $\mu_{\mathbf{\Delta}}(M) = 2$, and the normal vector c is chosen to be the gradient of the function $\overline{\lambda}_r(M\Delta)$ at the vertex p where the worst case is achieved. The curve marked by 'x' indicates the boundary where the singularity of $I - M\Delta$ occurs.

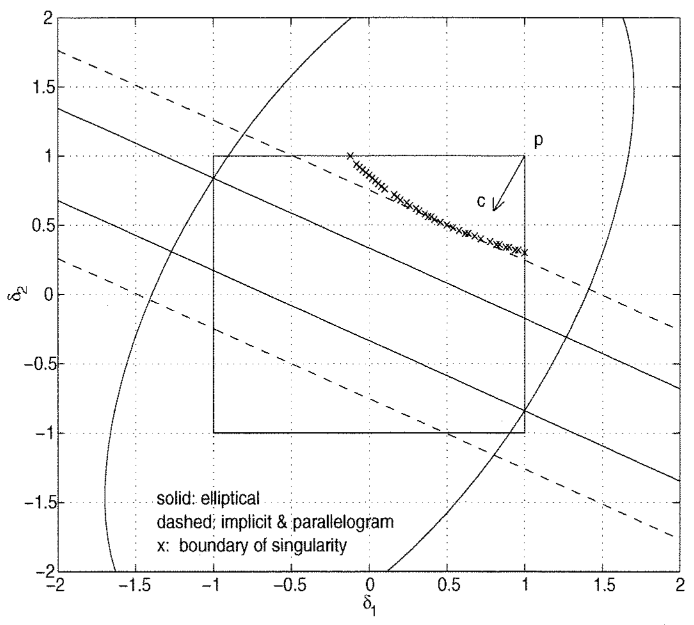

Figure 5.4: Upper bound for μ with linear cuts: Example 2.

Table 5.1 shows the results of applying the three methods for computing upper bounds to 150 random matrices of size 2, 3, and 4, with 50 matrices of each size. The problems are constructed so that 1 is a guaranteed lower bound on $\mu^s_{\mathbf{\Delta},lc}(M)$, but the actual value is unknown. So an upper bound close to 1 is known to be good, but no conclusions can be made for upper bounds that are far from 1. However, relative performance of the bounds can be observed by comparing the bounds obtained by different methods. For some problems, the parallelogram and ellipsoid method are unable to find any bound to $\mu^s_{\mathbf{\Delta},lc}(M)$. The entries in the column labeled '#' indicate the number of problems for which that method is able to compute an upper bound. The entries in the column labeled 'avg. ub' is the average upper bound computed for the problems on which a bound is obtained with the associated method. The implicit method gives by far the best bounds, especially as the problem size increases. The computation cost is not listed since different LMI solvers have been used. By experience the implicit and parallelogram methods are fairly compatible in terms of computation time, while the elliptical method is the most time-consuming because the algorithm used tries to solve a nonconvex optimization problem. The implicit is hands down the best of the three methods. Unfortunately, the absolute quality of the bounds remains unknown without a lower bound. The development of a lower bound for μ with linear cuts can be a subject for future research.

Size	$n = 2$		$n = 3$		$n = 4$	
Method	#	avg. ub	#	avg. ub	#	avg. ub
Implicit	50	1.0540	50	1.1320	50	1.4947
Parallel.	50	1.1882	50	1.6123	48	2.3534
Elliptical	50	1.2376	32	1.8136	22	3.4617

Table 5.1: Comparison of average upper bounds achieved by three different methods.

Chapter 6 Linear Cuts for Probabilistic μ — The LC Algorithm

Our objective is to compute hard bounds and soft bounds on purely probabilistic μ, since its exact computation is intractable. Direct application of the naive branch and bound algorithm with axially aligned cuts (the BNB algorithm) to computing bounds on probabilistic μ failed to give satisfactory solutions, as was shown in Chapter 4. The introduction of μ with linear cuts in the previous chapter provided an alternative by adding linear constraints to the set of parameters, which was motivated by the analysis of rank-one problems. In the next section the upper bound for μ with linear cuts is applied to compute a hard upper bound on purely probabilistic μ, which helps the computation of the soft bound as well. Numerical experiment results on various kinds of random matrices will be demonstrated afterwards.

6.1 Probabilistic μ Upper Bound using Linear Cuts

First let us quickly review the formulation of the purely probabilistic μ problem. The function $P(\gamma)$ defines the complementary cumulative distribution of certain performance function, in particular $\overline{\lambda}_r(M\Delta)$. When the parameter Δ is uniformly distributed in the unit box $\boldsymbol{B\Delta}$, evaluating $P(\gamma)$ is equivalent to accurately separating the region of bad events $\boldsymbol{R_b}$ from the rest of the parameter set. In the special case when M is rank-one, the separating surface $\boldsymbol{S}_\gamma$ becomes a hyperplane in $\mathbb{R}^n$. Then the best way to isolate $\boldsymbol{R_b}$ is to use a linear cut aligned with $\boldsymbol{S}_\gamma$. For problems that are not rank-one, the separating surface $\boldsymbol{S}_\gamma$ can be of any shape, in which case the linear cut will only be an approximation. The accuracy of this approximation needs to be tested via numerical experiments.

The idea of using linear cuts to compute an upper bound on $P(\gamma)$ follows. Suppose

$\mu_{\boldsymbol{\Delta}}(M) = 1$. If for a given $\gamma \leq 1$ the upper bound for the skew version of μ with linear cuts $\overline{\mu}^s_{\boldsymbol{\Delta},lc}(\frac{M}{\gamma}) \leq \beta$, then it is guaranteed that $\overline{\lambda}_r(M\Delta)$ on the region $\boldsymbol{B\Delta}^s_{lc,\beta} = \{\Delta \in \boldsymbol{\Delta} : \|\Delta\| \leq 1 \text{ and } |c^T\delta| \leq \frac{1}{\beta}\}$. Making no contribution to $P(\gamma)$, the region $\boldsymbol{B\Delta}^s_{lc,\beta}$ can be removed, and the volume of the remaining region will be an upper bound on $P(\gamma)$.

The following is an implementation of the above method. This algorithm will be referred to as "the LC algorithm." The hard upper bound and the soft bound computed are denoted by hub_{LC} and sb_{LC}, respectively.

1. For random matrix M, compute $\overline{\lambda}_r(M\Delta)$ on all the vertices of $\boldsymbol{B\Delta}$ to obtain $\mu_{\boldsymbol{\Delta}}(M)$ and the worst-case vertex P. Scale $M = \frac{M}{\mu_{\boldsymbol{\Delta}}(M)}$.

2. Compute the function gradient at P, denoted by g, and let $c = -\frac{g}{\|g\|}$, then c will be the unit normal vector of the cutting hyperplanes.

3. Compute k by singular value decomposition of M, construct the implicit system $(\tilde{M}, \tilde{C})$ with $\frac{M}{\gamma}$, k, and c.

4. Compute $\overline{\mu}^s_{\boldsymbol{\Delta},lc}(\frac{M}{\gamma})$ using implicit μ upper bound. Suppose the lowest bound achieved is β^*. Optimize over k if β^* is not good enough.

5. Cut $\boldsymbol{B\Delta}$ with the two hyperplanes $\{c^T\delta = \pm\frac{1}{\beta^*}\}$. Suppose the remaining two corners around P and $-P$ are $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$. Check the vertices in $\boldsymbol{R}_2$, if any of them are "bad", mark $\boldsymbol{R}_2$ as "bad" region as well as $\boldsymbol{R}_1$, and $hub_{LC} = \frac{V(\boldsymbol{R}_1)+V(\boldsymbol{R}_2)}{V(\boldsymbol{B\Delta})}$; otherwise only $\boldsymbol{R}_1$ is "bad" and $hub_{LC} = \frac{V(\boldsymbol{R}_1)}{V(\boldsymbol{B\Delta})}$.

6. Generate N random samples uniformly in the bad region, compute $\overline{\lambda}_r(M\Delta)$ on these samples, denote the number of bad samples as N_b, then $sb_{LC} = \frac{N_b}{N \cdot hub_{LC}}$.

The following technical issues need to be considered.

- Due to the special structure of the purely real μ problem, $\mu_{\boldsymbol{\Delta}}(M)$ can be computed by checking all the vertices of $\boldsymbol{B\Delta}$, which is only feasible for problems with small size because of the exponential growth in the number of vertices. For bigger problems the idea is using standard B&B first to identify potentially bad regions, then performing linear cuts to see if any bad corners can be isolated. This idea will be explored in the next chapter.

- The resulting hub_{LC} may be conservative due to the following problems:

  a) The bad region $\boldsymbol{R_b}$ contains multiple corners of $\boldsymbol{B\Delta}$ so that a single linear cut cannot isolate all of them at the same time. This may be an artifice of random matrices instead of a characteristic of real physical systems.

  b) The shape of the separating surface $\boldsymbol{S}_\gamma$ may be complicated even near the worst case such that a hyperplane is not a good approximation for it, although we expect most problems to be close to rank-one near their worst cases, as predicted as Chapter 4.

  c) The upper bound for μ with linear cuts is not tight enough.

- Problem a) and b) raise the question of what kind of matrices are more representative of the problems from the real world. Apparently purely random matrices are not very good in this sense. So the matrices used in our numerical tests are so-called "random decaying matrices", where each element $M(i,j)$ is a random floating point number with normal distribution $\mathcal{N}(0, \frac{4}{(i+j)^2})$, a slight modification of the 'randn' matrices in MATLAB. These matrices tend to be more rank-one alike and not to have distributed bad points, which may be more typical of physical problems.

- There are many ways to improve the bound, for instance, applying off-centered cuts or multiple linear cuts to capture bad regions more efficiently or to approximate the level set more accurately, or developing global optimization techniques to achieve a better bound for μ with linear cuts. And finally, if none of the above schemes work, branch and bound type of algorithms will be necessary to minimize the hard upper bound in an iterative fashion.

6.2 Numerical Experiments

6.2.1 Rank-one Matrices

Rank-one problems are the motivation of doing linear cuts for probabilistic μ. So random rank-one matrices were used to test the effectiveness of the LC algorithm. The upper bounds achieved are exact just as the theory predicts, i.e., $hub_{LC} = P(\gamma)$. This gives us hope for good quality of the bounds for problems that are close to rank-one near their worst cases.

6.2.2 Random Decaying Matrices

The LC algorithm was tested on 60 random decaying matrices. The ratio of the soft bound sb^d_{LC} to the hard upper bound hub^d_{LC} is denoted by rt^d_{LC}, where the superscript 'd' indicates the results was obtained on random decaying matrices. The ratios for all the problems are shown in Figure 6.1. Again set $\gamma = \eta \times \mu_{\mathbf{\Delta}}(M)$ and $\eta = 0.9$. For $n = 2$, rt_{LC} is mostly very close to 1; for $n = 4$, all the problems have the ratio above 0.1 except for one; for $n = 6$, 17 out of 20 problems have the ratio above 0.1. So the algorithm performs reasonably well for most problems. (See [79] for a typical example.) However, there always exist a few bad problems like the other 3 of size 6 for which the gap between sb_{LC} and hub_{LC} is too big for the bounds to be useful.

To compare the performance of the LC algorithm with the BNB algorithm on random decaying matrices, the BNB algorithm was also run on the same 60 problems as above. The results were obtained after branching 100 times. Let $rt^d_{BNB} = \frac{sb^d_{BNB}}{hub^d_{BNB}}$. Then the ratio of rt_{LC} to rt^d_{BNB} is shown in Figure 6.2. It is easy to see that the LC algorithm beats the BNB algorithm on most random decaying matrices, and the advantage becomes more significant as n increases. This result is not so surprising because random decaying matrices are similar to rank-one matrices in some sense, and the LC algorithm is specifically designed to deal with rank-one problems.

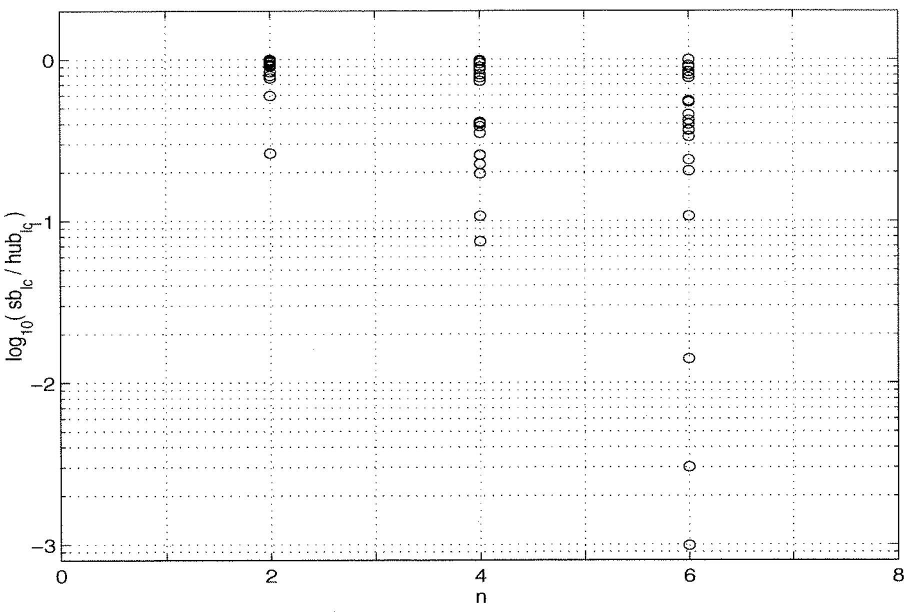

Figure 6.1: $rt^d_{LC} = \frac{sb^d_{LC}}{hub^d_{LC}}$ vs. problem size for 60 random decaying matrices, with 20 matrices of each size.

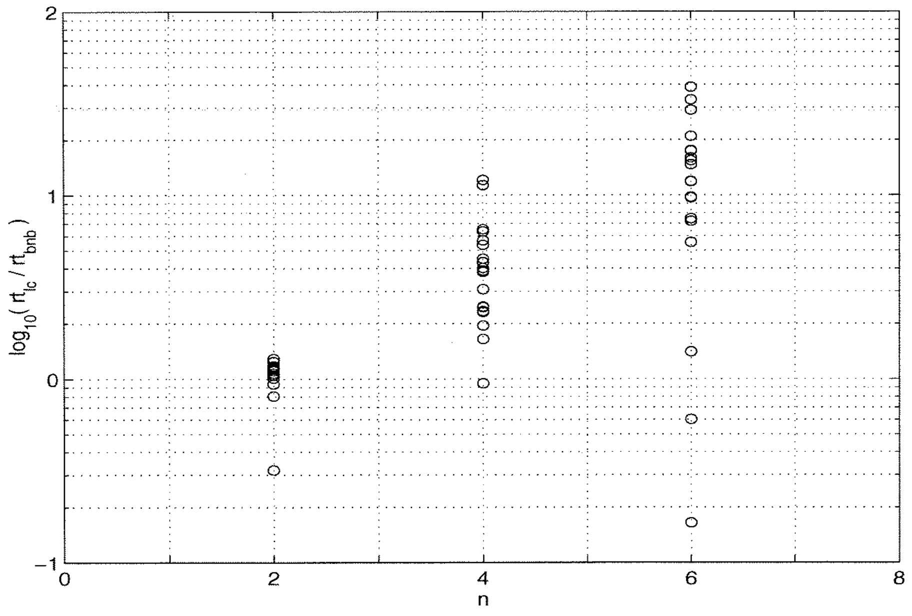

Figure 6.2: $\frac{rt^d_{LC}}{rt^d_{BNB}}$ vs. problem size for the same 60 random decaying matrices.

6.2.3 General Random Matrices

Since the LC algorithm implements only a single linear cut, it is not suitable for general random matrices for which $\boldsymbol{R_b}$ tends to spread out in the parameter set $\boldsymbol{B\Delta}$, which was verified by testing the LC algorithm on the same 80 random matrices tested under the BNB algorithm. Again set $\eta = 0.9$. Except for $n = 2$, there are always problems for which the LC algorithm failed to get any hard bounds, because a single linear cut cannot separate $\boldsymbol{R_b}$ from $\boldsymbol{R_g}$. The number of problems for which the algorithm was able to compute a hub_{LC} is 20, 14, 4 and 4 for size 2, 4, 6 and 8, respectively. And the average rt_{LC} for these problems are 0.9265, 0.3256, 0.0120 and 0.0021, respectively, slightly better than the average ratios achieved using the BNB algorithm. The intuition is that each of these problems has only one bad corner in $\boldsymbol{B\Delta}$, which can be isolated using a single linear cut. In the neighborhood of this bad corner, the problem is closer to rank-one so that a hyperplane is a relatively better approximation of the separating surface than the boundary of a union of hypercubes. However, even for these relatively "easier" problems the gap between hub_{LC} and sb_{LC} increases very quickly with the problem size. Although random matrices may not be good representatives of real physical systems, an algorithm that suits not only rank-one like problems is desirable, which will be the focus of the next chapter.

Chapter 7 The Mixed LC-BNB Algorithm

In the previous chapters we described two methods that have been used to compute hard upper bounds and soft bounds for probabilistic μ, in particular, $P(\gamma)$. The BNB algorithm is straightforward to understand and implement. The resulting branches are hyperrectangles in $\mathbb{R}^n$. It is easy to keep track of them, to compute their volumes, and to generate random samples in them. But it is not an efficient way to approximate a complex surface in a high dimensional space unless this surface happens to be aligned with the axes. On the other hand, a linear cut approximates the separating surface $\boldsymbol{S}_\gamma$ with a hyperplane, which is more efficient if the problem is close to rank-one. However, for the problems that are far from rank-one, either multiple bad corners exist or the shape of $\boldsymbol{S}_\gamma$ is complicated, then multiple linear cuts would be required. But one linear cut on top of another is hard to track analytically. Moreover, before implementing the linear cut, the worst vertex needs to be identified, which involves worst-case μ computation, where a branch and bound type of algorithm is often required to get a sufficiently accurate answer. Although the BNB algorithm alone is not enough to provide a good hard upper bound on $P(\gamma)$, it does reveal some information about the distribution of the bad events in the parameter space during the branching process. Therefore, it is natural to combine the LC algorithm with the BNB algorithm to compute bounds for $P(\gamma)$ [80].

7.1 The Algorithm

The idea is to take advantages of both algorithms by using a simple branch and bound scheme with axial cuts to compute bounds on $\mu_{\boldsymbol{\Delta}}(M)$, divide $\boldsymbol{B\Delta}$ into smaller regions to separate bad corners from each other, while using the LC algorithm to compute a hard upper bound (hub) and a soft bound (sb) for $P(\gamma)$ for each region. There are two objectives: first, the global upper bound (U) and lower bound (L) on $\mu_{\boldsymbol{\Delta}}(M)$ are close enough, i.e., $\frac{U-L}{U} \leq TOL$; second, the ratio of sb to hub on $P(\gamma)$ is above $RTOBJ$. Keep branching until both goals are reached, or the step number reaches $MAXSTEP$. The resulting algorithm is called the LC-BNB algorithm. The following parameter values are used: $TOL = 0.01$, $RTOBJ = 0.95$, $MAXSTEP = 200$, $\eta = 0.9$.

1. Initialization: Let $\boldsymbol{B}_0 = \boldsymbol{B\Delta} = \boldsymbol{B}(0,1)$, $v_0 = V(\boldsymbol{B}_0) = 1$.

2. Compute upper bound (ub_0) and lower bound (lb_0) for $\mu_{\boldsymbol{\Delta}}(M)$ on $\boldsymbol{B}_0$ without branching, which also gives a candidate for the worst vertex P_0. Let $U = ub_0$, $L = lb_0$, and $\gamma = \eta L$.

3. Call the LC algorithm to compute hub_0 and sb_0 for $P(\gamma)$. If $rt_0 = \frac{sb_0}{hub_0} \geq RTOBJ$ and $\frac{U-L}{U} \leq TOL$, stop; otherwise, define the set of branches $\boldsymbol{S_B} = \{\boldsymbol{B}_0\}$, $step = 0$.

4. Branching: $step = step + 1$. If $\frac{U-L}{U} > TOL$, pick $\boldsymbol{B} \in \boldsymbol{S_B}$ with the maximum ub; otherwise, pick $\boldsymbol{B} \in \boldsymbol{S_B}$ with the minimum $\frac{rt}{hub \cdot v}$. Choose a parameter to cut, divide $\boldsymbol{B}$ by half into $\boldsymbol{B}_1 = \boldsymbol{B}(\Delta_1, r_1)$ and $\boldsymbol{B}_2 = \boldsymbol{B}(\Delta_2, r_2)$. Make sure that $\mu(M, \Delta_1) < \gamma$ and $\mu(M, \Delta_2) < \gamma$; if not, change the location of the cut until the above criterion is satisfied. Compute $v_i = V(\boldsymbol{B}_i)$, $i = 1, 2$. $\boldsymbol{S_B} = \boldsymbol{S_B} \setminus \{B\}$.

5. For $i = 1, 2$, compute μ upper and lower bounds in $\boldsymbol{B}_i$, get ub_i, lb_i, P_i. Update U, L, and γ.

6. For $i = 1, 2$, if $ub_i < \gamma$, throw away $\boldsymbol{B}_i$; otherwise, call the LC algorithm to compute hub_i and sb_i in $\boldsymbol{B}_i$. If $hub_i = 0$, throw away $\boldsymbol{B}_i$; otherwise, $rt_i = \frac{sb_i}{hub_i}$, $\boldsymbol{S_B} = \boldsymbol{S_B} \cup \{\boldsymbol{B}_i\}$.

7. Compute the current overall ratio of sb to hub: $rt = \frac{\sum_i sb_i \cdot v_i}{\sum_i hub_i \cdot v_i}$. If $rt >= RTOBJ$ and $\frac{U-L}{U} \leq TOL$, go to step 8; otherwise, go to step 4.

8. Compute the overall hard upper bound $hub_{LC-BNB} = \sum_i hub_i \cdot v_i$ and determine the total sample size N. For each $\boldsymbol{B}_i \in \boldsymbol{S_B}$, sample the corner isolated by the

linear cut with sample size $np_i = hub_i \times v_i \times N$, calculate sb_i, then the overall soft bound $sb_{LC-BNB} = \sum_i sb_i \cdot v_i$.

Some technical details are left out since they are not crucial elements of the algorithm. One thing to be pointed out is that during the branching process, each call to the LC algorithm involves a sampling in the corresponding branch, which is just an intermediate step to estimate sb for that branch so that the next step can pick the particularly problematic branch to cut. A fixed small sample size $np = 100$ is used for this estimation in each step so that the computation time is reasonable. In the end, resampling is required since to make the samples uniformly distributed in $\boldsymbol{B\Delta}$, the sample size for each isolated region should be proportional to its volume, as implemented in Step 8. And since a large portion of the parameter set has been excluded by the hard bound computation, a relatively small sample size in the potentially bad regions $\tilde{\boldsymbol{R}}_b$ is equivalent to an effective sample size several orders of magnitude larger in the whole set $\boldsymbol{B\Delta}$, which makes it possible to track the probability of extremely rare events.

7.2 Numerical Experiments

The test results of the LC-BNB algorithm on the same 80 random matrices as those used for the BNB and the LC algorithms are displayed in Figure 7.1 and Figure 7.2. On all the problems the LC-BNB algorithm was able to compute a hard upper bound and a positive soft bound. The average number of steps taken is 0.5, 64.3, 193.9 and 200 for size 2, 4, 6 and 8 respectively. For $n = 2$ the mean step number is smaller than 1 since for most problems a single linear cut is enough and so no branching is required, while for $n = 8$ rt_{LC-BNB} never achieves the objective 0.95, so for every problem the step number reaches the maximum, which is 200. In Figure 7.1 hub_{LC-BNB} follows quite closely the decay of sb_{LC-BNB} with the problem size, which means the algorithm was able to identify most of the benign regions and exclude them from the sampling. Figure 7.2 shows the ratio of the two bounds for each problem versus the problem size. The average rt_{LC-BNB} is 0.9938, 0.9306, 0.6402 and 0.2471 for size 2, 4, 6 and

8, respectively. For $n = 2$ all the problems achieve a ratio higher than 0.99 except one problem, for which the ratio is 0.94. For $n = 8$ all the problems get a ratio higher than 0.1 except for 3 of them. So the performance of the algorithm does degrade when n increases, but the decay is much slower than for the previous two methods.

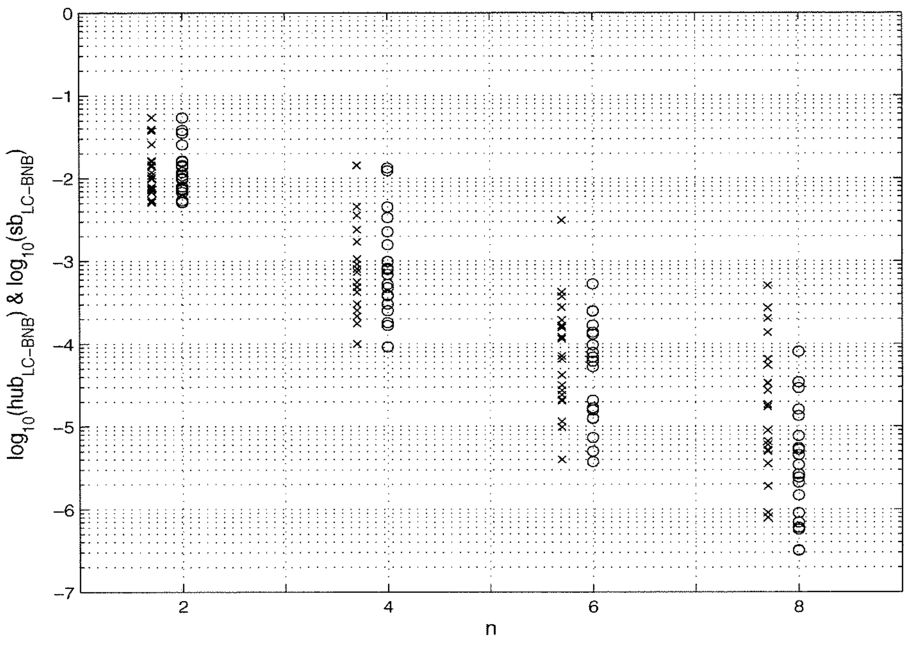

Figure 7.1: hub_{LC-BNB}('x') and sb_{LC-BNB}('o') for 80 general random matrices.

7.3 Comparison of Three Methods

To compare all three methods (BNB, LC and LC-BNB), the average ratios achieved with the three different algorithms versus problem sizes are displayed in Table 7.1. The higher the ratio, the tighter the hard upper bound and the soft bound. Note that the average is calculated on those problems for which the corresponding ratio is nonzero, which happens when a hard upper bound is available and the soft bound is positive. The numbers of such problems are listed in the column labeled '#' under each problem size. It is easy to see that the average performance of the mixed LC-BNB algorithm is much greater than the other two, in terms of the number of problems for which the algorithm was able to compute useful bounds, the absolute values of

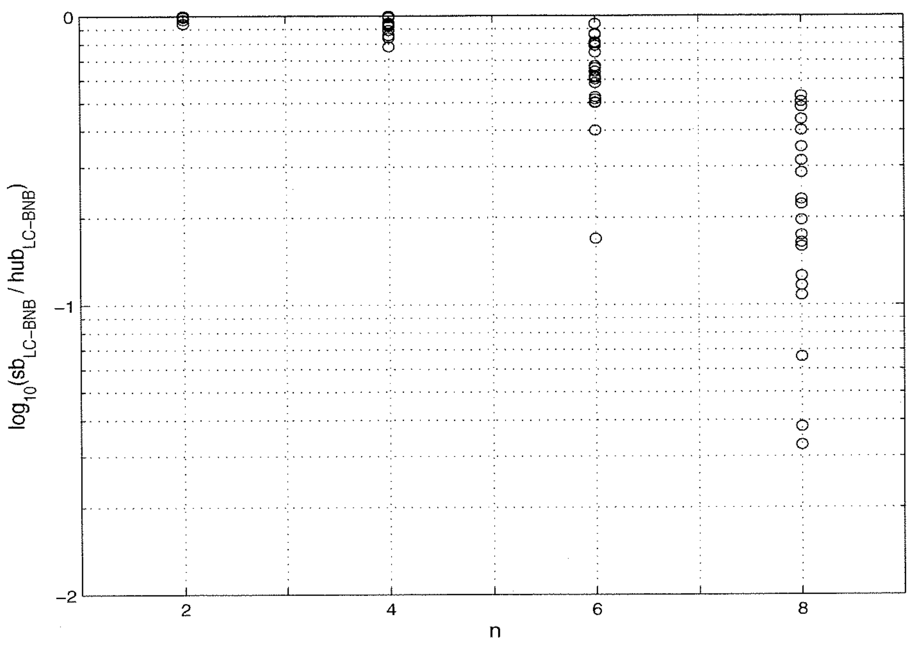

Figure 7.2: $rt_{LC-BNB} = \frac{sb_{LC-BNB}}{hub_{LC-BNB}}$ for the same 80 random matrices.

the ratios which indicate the tightness of the two bounds, and the relative decay of the performance with the size of the problem. The intuition behind the effectiveness of the LC-BNB algorithm follows. The use of the axially aligned branching scheme divides the parameter set into smaller regions to improve the worst-case μ and the probabilistic μ computation at the same time. Meanwhile, if the region of bad events $\boldsymbol{R_b}$ spreads out in the whole parameter space, it will be contained in different branches so that one does not interfere with another, which makes it possible for a single linear cut to isolate the bad corner in each branch. If this is not feasible for some branch, the branching simply continues, trying to turn the complex branch into easy ones. The same ratios are again shown in Figure 7.3, which is a clearer demonstration of the relative performance of these three algorithms.

Size	$n=2$		$n=4$		$n=6$		$n=8$	
Algorithm	#	avg. rt	#	avg. rt	#	avg. rt	#	avg. rt
BNB	20	0.8001	20	0.1986	20	0.0090	18	0.0004
LC	20	0.9265	14	0.3256	4	0.0120	4	0.0021
LC-BNB	20	0.9938	20	0.9306	20	0.6402	20	0.2471

Table 7.1: Comparison of average ratios obtained by three different algorithms.

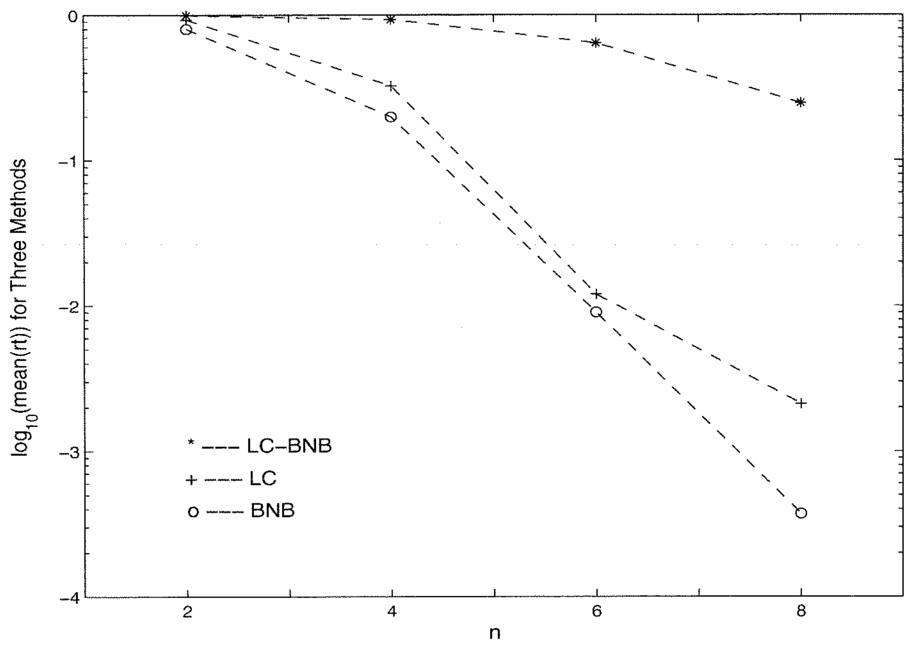

Figure 7.3: Average $rt = \frac{sb}{hub}$ vs. problem size with three different algorithms.

7.4 The LC-BNB Algorithm vs. Standard Monte-Carlo

After examining the relative performance of the three methods for computing bounds on probabilistic μ and concluding that the LC-BNB algorithm is the best based on numerical experiment results, we may want to go back and ask the question that how is the performance of the LC-BNB algorithm compared to that of standard Monte-Carlo simulations. Although Monte-Carlo simulations do not provide hard bounds that can be given using the LC-BNB algorithm, the former has the advantage of being easy to implement and probably requiring less computation to get a similar estimate. Here we use one particular example to illustrate that this is not the case when the probability to be estimated is extremely small.

The following is a random matrix of size 6:

$$M = \begin{bmatrix} 0.9442 & 1.5210 & 0.5869 & 2.3093 & 0.4855 & 0.1034 \\ -2.1204 & -0.0384 & -0.2512 & 0.5246 & -0.0050 & -0.8076 \\ -0.6447 & 1.2274 & 0.4801 & -0.0118 & -0.2762 & 0.6804 \\ -0.7043 & -0.6962 & 0.6682 & 0.9131 & 1.2765 & -2.3646 \\ -1.0181 & 0.0075 & -0.0783 & 0.0559 & 1.8634 & 0.9901 \\ -0.1821 & -0.7829 & 0.8892 & -1.1071 & -0.5226 & 0.2189 \end{bmatrix}. \tag{7.1}$$

The LC-BNB algorithm was run for 200 steps, which required approximately 6.39×10^8 flops (floating point operations) in computation. The upper and lower bounds achieved on $\mu_\Delta(M)$ were $U = 2.2032$ and $L = 2.2027$, respectively. Together they guarantee that the performance in the worst case is as least as bad as 2.2027 but definitely no worse than 2.2032. The error between these two bounds is within 0.013% of the lower bound, which is small enough for practical purposes. So we can take the lower bound as the value of the worst performance. Suppose this value is really bad for this particular system, we would like to know how probable it is. In the other words, we want to estimate the probability of the performance function being near the worst-case. Since this question cannot be answered by the worst-case computation, a naive Monte-Carlo sampling algorithm was run to estimate $P(\gamma)$ with $\gamma = 0.9{*}2.2027 = 1.9824$. Out of the $100,000$ samples tested, 10 of them hit the region of bad events $\boldsymbol{R_b}$, so the estimate of $P(\gamma)$ is 1×10^{-4}. The number of flops required for this estimate was around 8.36×10^{8}. So the Monte-Carlo simulation tells us that bad events are highly unlikely to occur. However, the confidence level with respect to 10% accuracy calculated using the Chernoff bound [17] is only 0.002%, which basically means no confidence. This is because the probability we want to estimate is so small that a much larger sample size is required to get a useful confidence level. In fact, to achieve 80% confidence that the real probability is within $\pm 10\%$ of 1×10^{-4}, the required sample size is around 8.05×10^{9}, which is equivalent to approximately 6.73×10^{13} flops of computation, or approximately $12,000$ hours of CPU time on a SUN Ultra workstation. At the same time, the LC-BNB algorithm gave a hard upper bound of 1.19×10^{-4} and a soft bound of 9.57×10^{-5} on $P(\gamma)$ in about 12 minutes. The hard bound guarantees that $P(\gamma)$ is definitely below 1.19×10^{-4}, which is fairly small in most situations. The soft bound is about 80% of the hard upper bound, with a confidence level of virtually 100% that the real value of $P(\gamma)$ is within $\pm 10\%$. Besides the benefit of having a much larger effective sample size (8.37×10^{7}) in the whole parameter set $\boldsymbol{B\Delta}$ by excluding most of the benign region (99.988% of $\boldsymbol{B\Delta}$), sampling only in a small subset of $\boldsymbol{B\Delta}$ leads to a lower variance in the resulting estimate. This example clearly demonstrates that in the case of estimating the probability of extremely rare events, the LC-BNB algorithm is much more effective than a naive Monte-Carlo simulation.

Chapter 8 Concluding Remarks

To conclude, we first summarize the main results in this thesis, then discuss possible future research directions.

8.1 Summary

This thesis explores the computational complexity involved in computing both hard and soft bounds in probabilistic robustness analysis. As an extension to the standard μ framework reviewed in Chapter 2, probabilistic μ was defined in Chapter 3 as some probability measure of a system's stability or performance, assuming that the probability distribution on the uncertainty associated with the system is given. This general framework was further simplified to purely probabilistic μ, which only concerns real parametric uncertainty, but we believe still reveals the intrinsic difficulty associated with more general settings. Faced with the intractability implied by the NP-hardness of the exact computation, branch and bound type of methods are needed to obtain approximations of purely probabilistic μ. Three different algorithms have been investigated in this thesis.

The BNB algorithm introduced in Chapter 4 was a direct application of a naive branch and bound algorithm that has proven successful in worst-case μ computation. However, it did not claim the same success in computing bounds for probabilistic μ. Unlike for the worst-case, for probabilistic computation, exploring more intelligent branching schemes seems to be more critical than improving the quality of the original bounds. The intuition is that exponential growth in the number of branches is inevitable if the branching is not appropriately aligned with the separating surface of the good and bad regions in the parameter space. As one extreme of the space of examples, rank-one problems have a separating surface being a hyperplane, which would be much better captured by a linear cut in the uncertain parameters. This observation was the motivation to develop the second algorithm — the LC algorithm.

The machinery needed for performing linear cuts in the parameter space is μ with linear cuts, a framework that permits linear constraints on the uncertain parameters, which was reviewed in Chapter 5. Among the three upper bounds on μ with linear cuts that were discussed, the upper bound by the implicit method demonstrated the best average quality on numerical examples. This was the basis for the LC algorithm presented in Chapter 6 for computing a hard upper bound on probabilistic μ. The use of a single linear cut in the LC algorithm determines that it is only suitable for problems that have only one bad corner in the parameter set and are close to rank-one in the neighborhood of this bad corner. For problems that have multiple bad corners or have complicated shaped separating surface, more sophisticated branching schemes are required to isolate the bad region from the rest of the parameter set.

Chapter 7 explored the above idea and proposed a new algorithm based on the two existing algorithms. It was is referred to as the mixed LC-BNB algorithm. It uses standard branch and bound algorithms with axially aligned cuts to divide the parameter set in order for the bad corners to be separated from each other so that a linear cut in each branch can isolate the bad corner easily. This algorithm combined the advantages of both the BNB and the LC algorithms in producing tighter hard and soft bounds for probabilistic μ. The effectiveness of this combination has been demonstrated by numerical experiments. Compared to standard Monte-Carlo simulations, the LC-BNB algorithm greatly reduces the variance in the resulting estimates by excluding benign regions and increasing effective sample size.

8.2 Future Directions

The results in this thesis are far from being conclusive and there are many open questions which can be the subjects of future research.

As suggested by the numerical experiment results, even with the mixed LC-BNB algorithm, the ratio of the soft bound to the hard upper bound decays fairly quickly with the size of the problem. It is foreseeable that the performance will become even worse when the problem size keeps increasing. To get tolerable performance for larger problems, the computation effort involved will be much greater. It remains an open question how to deal with these higher dimensional problems with reasonable growth rate of the computation.

Another interesting direction is how to extend this type of computation from the simplest setting addressed in this thesis to more general frameworks. Many possible extensions exist. The most straightforward one is to the case when the real parameters are repeated, or when the matrix M is not real. In either case, the fact that the worst-case is achieved on vertices ceases to hold, but the parameter space stays the same and branch and bound algorithms can still be applied to get similar bounds on probabilist μ. Another possibility is to allow probability distributions on complex scalars and complex full blocks, which will result in a tremendous increase in the dimension of the parameter space. If a branch and bound algorithm is to be implemented on these parameters, the number of branches blows up very quickly without necessarily significant improvement of the computation, therefore this extension seems less attractive from a practical point of view. The more interesting case is when the mixed probabilistic μ is concerned, for instance, probabilistic performance with guaranteed stability. We may be able to apply similar branch and bound techniques as the ones we used for the purely probabilistic μ to compute hard bounds on the mixed probabilistic μ. But the computational complexity involved needs further investigation through extensive numerical experiments.

Most importantly, it is desirable to apply the probabilistic μ framework to robustness analysis of more physically motivated problems. Random matrices in general may not be good representatives of real physical systems. It will be beneficial to take advantage of special structures present in many engineering systems so that more efficient computation schemes can be developed.

Part II

Generalized Source Coding and Optimal Web Layout Design

Chapter 9 Introduction

The emergence of the Internet over the past twenty years has turned computer networks from sets of locally connected host computers and terminals into a globally distributed information exchanging and processing system. To people throughout the world, the influence of the Internet is tremendous. Communications become easier and faster, information stored in a fragmented manner is now accessible to a global audience, and businesses are conducted in new and more efficient ways. Personal computers have evolved from pure computing devices sitting on people's desks, into windows of communication that connect each individual to the rest of the world. All of these are enabled by the increasing power of the Internet.

On the other hand, as a large-scale, heterogeneous, complex interconnected system, the Internet demonstrates many features that are unparalleled by other engineering systems, and confronts network researchers with immensely challenging problems. One such feature is the unprecedented growth rate in the scale of the Internet. While there were only about 1.3 million Internet hosts in January 1993, the number increased to approximately 72.4 million in January 2000. It has been increasing exponentially with a growth rate of around 80%/year,[1] and there is no evidence indicating that this growth will slow down in the near future. Another important feature of the Internet is its enormous heterogeneity that exists on many levels, including the underlying physical links that carry the information, the protocols that inter-operate over the links and manage the traffic, the mix of applications run at end systems, and the degrees of congestion at different times and locations. Moreover, the dramatic change of the Internet over time lies not only in its scale, but also in the ways it is used. Increasing processor speed and network bandwidth continue spurring development of new applications, such as, multimedia, real-time video and audio, distributed computing, multi-player network gaming, and the World Wide Web. In the meantime,

[1]Source: Internet Software Consortium (http://www.isc.org/)

the ascending popularity of these applications has in turn driven the need for more powerful computers and higher speed networks. In particular, the World Wide Web, or the Web for short, originally prototyped in 1991, has quickly become the dominant force that is driving the explosive growth of the Internet [61].

The enormous number of users and rich class of applications on the Internet have made its robustness a highly desirable property, even more important than its efficiency. We have all experienced system break downs, "requested URL not found", broken pipes, slow downloads, and so on. In particular, for the Web users, a survey in 1997 showed that slow access and inability to find relevant information are the two most frequently reported problems [62]. And network congestion is at least partially responsible for the slow access problem. Congestion control has become the most challenging task that is facing network researchers today. It has many facets, including modeling of Internet traffic, generation of synthetic traffic traces that can be used in simulations, analysis of queueing and network performance under different traffic patterns, and design of network protocols both for the end systems and at the network level. All these problems are intimately related. As a consequence, understanding the characteristics of network traffic becomes more and more critical to designing robust and reliable networks and network services.

Since WWW connections currently dominate Internet traffic, the Web traffic is chosen as the object of study in this thesis. The goal is to provide a plausible physical explanation for the empirically observed characteristics in measured Web traffic based on the theory of *robust design* of uncertain systems.

9.1 Related Work

High quality measurements have been carried out on traffic in various computer networks since early 1990s. Statistical analysis on the empirical data has shown strong indication of the *self-similar* nature of the traffic in both local area networks(LANs) [41] and wide area networks(WANs) [60], quite unlike the traditionally assumed Poisson traffic models. This means that real traffic data exhibits long-range dependence and high burstiness over a wide range of time scales. In contrast, Poisson models have a characteristic burst length that would be smoothed out by averaging [72].

These discoveries have inspired extensive and interesting research in the modeling of network traffic, its relationship to network protocols and its impact on network performance [42, 59, 37, 21, 3]. For instance, analysis and simulation results have shown a significant difference in queueing performance between traditional traffic models and self-similar models [55, 13, 26, 28]. As a consequence, ignoring the self-similar nature of the traffic at the modeling stage may lead to overly optimistic performance predictions and thus cause potential problems for buffer design and admission control in real high speed networks.

Moreover, evidence was presented in [19] that the subset of network traffic due to WWW transfers, recorded using NCSA Mosaic in early 1995, also demonstrated characteristics of self-similarity. Based on a mechanism of constructing self-similar traffic using a large number of ON/OFF sources that have period lengths drawn from *heavy-tailed* distributions [45, 73, 68], the authors in [19] traced the statistical properties of WWW traffic back to the distribution of Web transmission times, which in turn, is closely related to the distribution of the sizes of files transferred. Both distributions from measurement data exhibit heavy tails which decline like power laws with exponents close to 1. A newer data set collected in the same computing environment in 1998 exhibited similar characteristics, except that the power law tail became relatively lighter with an exponent at approximately 1.4 [9].

Although the recognition of heavy-tailed distributions in Web traffic was relatively new, there exists a rich body of literature on power laws in many complex bio-/eco-/techno-/socio-logical systems (CBETS). They have been found in power outages, forest fires, deaths and dollars lost due to man-made and natural disasters, income and wealth distribution of individuals and companies, variations in stock prices and federal budgets, and many other phenomena (see [14, 15] and the references therein). In addition to Web file transfers, more examples of heavy tails in networks and computer systems have been discovered, including sizes of files in a file system [35], CPU times consumed by UNIX processes [40], inter-keystroke times for typing [21], frame sizes for variable-bit-rate video [32], TELNET inter-packet times and lengths of FTP bursts [60], and sizes and durations of bursts and idle periods of individual Ethernet connections [48]. This ubiquity of power laws has motivated some researchers to suggest that they have a common origin in self-organized criticality [6]. It has also been suggested that it is possible to describe many of the features of computer networks in terms familiar in information theory, statistical physics, and the "new science of complexity" such as information, entropy, phase transitions, fractals, self-similarity, power laws, chaos, and so on. The hope is that these insights may lead to new approaches to network protocol design.

Ideas such as self-organized criticality have received much promotion in the last decade, but there have been as yet no convincing examples outside of carefully constructed laboratory experiments. Furthermore, while the "new science of complexity" has provided intriguing metaphors and popularized the notion that there are limits to reductionism, there has been little deep theoretical insights or practical applications. Recently, Carlson and Doyle [14] introduced a radically different theory for the nature of complexity and the origin of power laws and "phase transitions" in complex systems called Highly Optimized Tolerance (HOT). HOT systems arise when deliberate robust design aims for a specific level of *tolerance* to uncertainty, which is traded off against the cost of the compensating resources. *Optimization* of this tradeoff may be associated with some mixture of explicit planning as in engineering, or mutation and natural selection as in biology, but the word "design" is used loosely to encompass both. HOT systems in biology and engineering share many common features, including (1) high efficiency, performance, and robustness to uncertainties the systems are designed to deal with, (2) potential hypersensitivity to design flaws and unanticipated perturbations, (3) nongeneric, specialized, structured, and modular configurations, and (4) power laws.

The most important feature of the HOT theory for this thesis is in providing an alternative explanation for the origin of power laws in Web traffic. The connection between HOT and WWW files was first discussed in [14], and the connections with source coding were first made explicit in [25], which also looked at forest fires and compared HOT with standard SOC results. In this thesis a variety of more detailed models of varying tractability will be introduced, which present a consistent and coherent application of the HOT ideas to Web file layout. This approach, hopefully, can be applied to study power laws in other HOT systems as well.

9.2 Outline of Part II

Chapter 10 begins with a review of the nature of self-similar random processes and heavy-tailed distributions as a foundation for understanding the basic statistical framework behind self-similar network traffic. It then takes a further look into the previous work that established the connection between self-similar aggregate traffic and individual ON/OFF sources whose periods exhibit heavy-tailed distributions [73, 68]. The most relevant work to this thesis was the application of the above theory to WWW traffic and the observation of a similar connection in empirical data [19]. This study emphasized the importance of the heavy tails in Web file transfers, because they can be used to explain the self-similarity in WWW traffic, the largest contributor to the overall Internet traffic nowadays.

Now comes the central question: Where do heavy-tailed distributions of WWW file transfers come from? To answer this question, this thesis takes the basic approach of viewing the design of Web layout as a source coding problem, much like in standard information theory [18, 66], but with a new twist. A classical application for source coding is data compression. The model introduced in [25] that captures important elements common to both data compression and WWW design is reviewed in Chapter 11. It is referred to as the PLR model because it contains probabilities(P), losses(L) and resources(R). The interesting part is: what makes the Web layout design different from data compression? In data compression the aim in the design is to minimize the average length of codewords in order to reduce the cost of storage or transmission, subject to the constraint of decodability. Similarly, an important goal in Web layout is to minimize the delay in download times and latency, but the design variable is not the selection of codewords, but is instead the layout of the Web site itself. This would of course be in addition to any data compression that would be done to individual files. Such layout can be optimized for a given distribution of user interest, and a constraint, say, on the total number of files. As expected, average download times are minimized by having the high hits be small files, allowing larger files for rarely requested portions of the Web site. The resulting optimal distributions are very unlike those in standard Shannon theory and exhibit power law tails. The comparison between the prediction from the PLR model and the 1995 data is also shown in Chapter 11.

While the above PLR model provides some insights into the origin of power law distributions, it requires several rather unrealistic assumptions, which can be relaxed in various ways with some loss in the transparency of the solutions. With this spirit a more general PLR model is introduced at the beginning Chapter 12. It is this general model and its application to Web layout design that are the focus of this thesis. With the general PLR model the optimization problem is typically combinatoric. The global optimum can be achieved analytically or numerically on a special and simplified setting, or suboptimal solutions can be obtained using heuristics on more complex models. Chapter 12 takes the first approach, while the second approach is pursued in Chapter 13. We try to connect to the 1995 data first in Chapter 12. In 1995 the Web was in a nascent form, where much of the Web content was the result of putting preexisting documents on the Web. Since most such documents are essentially one-dimensional objects, Web layout in this context can be thought of to a first approximation simply as the problem of chopping up a document into pieces with links between. Design of such a Web layout can again be formulated as a generalized source coding problem, but the model is slightly more realistic than the PLR model with some additional features of a real Web site. However, focusing initially on the simple setting of one-dimensional documents that do not make extensive use of hyperlinks keeps the model still analytically tractable. The results are quite consistent with the simpler PLR model, as well as with the 1995 data.

In Chapter 13 we explore numerically the changes that would be expected as documents are designed specifically for the Web, including more complex layout for Web sites, more effective use of hyperlinks, and more internal structure in individual Web pages. To maximize the throughput, such a Web layout can be modified by splitting or merging files, with a tradeoff between ease of navigation, which would favor fewer files, and having small files to download. A heuristic optimization on random graphs is formulated, with user navigation modeled as Markov chains. Simulations on different classes of graphs always suggest that a Web site optimally designed to minimize the average latency a user experiences in browsing leads to power law distributions in the sizes of file transfers. In addition, the exponents of the power law tails are usually higher than those from the one-dimensional model, which is consistent with the difference found between the 1995 and 1998 data sets. Possible improvements on the graph-based Web layout models are proposed at the end of the chapter.

Finally, Chapter 14 summarizes the main results and suggests future research directions.

Chapter 10 Self-Similar Network Traffic and Heavy-Tailed Distributions

Measured network traffic can be characterized by discrete-time random processes, or time series $\boldsymbol{X} = \{X_k\}_{k=1}^{\infty}$. Typically the time line is uniformly divided into successive, nonoverlapping intervals of period Δt, and each random variable X_k represents the number of event occurrences during each time interval. For a *Poisson process* where independent events occur at random instants of time at an average rate of λ events per unit time, the above $\boldsymbol{X}$ is the increment process of the corresponding Poisson counting process with period Δt. All X_k are independent and identically distributed (i.i.d.) with

$$P[X_k = n] = \frac{(\lambda \Delta t)^n}{n!} e^{-\lambda \Delta t}, \qquad n \geq 0. \tag{10.1}$$

The Poisson process model has been successfully applied to describe call arrivals in the public switched telephone networks (PSTN) for at least fifty years, which led to the development of modern queueing theory for performance analysis and capacity planning. However, the Poisson framework is no longer valid for modeling traffic in the quickly emerging data networks, which exhibits high *burstiness* over a wide range of time scales. Instead, self-similar process models were proposed by many researchers to characterize traffic traces from a variety of packet-switching networks, including LANs [41], WANs [60], and WWW [19].

To understand self-similar network traffic and its possible causes, the next few sections review the theory of self-similar random processes and α-stable distributions, as well as the notion of heavy-tailed distributions, which attempts to draw a clear picture of the connections between self-similarity and power laws. All the theorems and lemmas are given without proofs. See [65] for an extensive and in-depth analysis.

10.1 Self-Similar Random Processes

The definition of *self-similarity* varies with the class of random processes it deals with. The standard one is for continuous-time nonstationary processes that have stationary increments, which will not be discussed here. The one that is more relevant to the traffic sequences defined above is for stationary time series. The define it, we need to first define *renormalization group transformations.*

Definition 10.1 *Let $\boldsymbol{X} = \{X_k\}_{k=1}^{\infty}$ be a discrete-time random process. Fix a number $H > 0$. For any $n \geq 1$, define the transformation $T_n : \boldsymbol{X} \rightarrow T_n\boldsymbol{X} = \{(T_nX)_k\}_{k=1}^{\infty}$, where*

$$(T_nX)_k = \frac{1}{n^H} \sum_{i=(k-1)n+1}^{kn} X_i, \qquad \textit{for all } k \in \mathbb{N}. \tag{10.2}$$

The transformations $T_n, n \geq 1$, are called **renormalization group transformations** *with critical exponent H.*

The family of transformations $\{T_n\}_{n=1}^{\infty}$ forms a semi-group since $T_{mn} = T_mT_n$, with T_mT_n being the transformation T_n followed by the transformation T_m. It is called the *renormalization group with exponent H*. A time series $\boldsymbol{X} = \{X_k\}_{k=1}^{\infty}$ is said to be a *fixed point* of the renormalization group if for all $n \geq 1$,

$$T_n\boldsymbol{X} \overset{d}{=} \boldsymbol{X}, \tag{10.3}$$

where "$\overset{d}{=}$" denotes equality in finite-dimensional distributions.

Definition 10.2 *A discrete-time, zero-mean, stationary random process $\boldsymbol{X} = \{X_k\}_{k=1}^{\infty}$ is called (exactly)* **self-similar** *or* **fractal** *with scaling parameter H if it is a fixed-point of the renormalization group $\{T_n\}_{n=1}^{\infty}$ with exponent $H \in [0.5, 1)$. It is said to be* **asymptotically self-similar** *if (10.3) only holds as $n \rightarrow \infty$.*

The scaling parameter H is a measure of the degree of self-similarity. The higher the H, the "more" self-similar the process. The reason will become clear in the following example.

Example 10.3 *Fractional Gaussian noise: A stationary Gaussian random process* $\boldsymbol{X} = \{X_k\}_{k=1}^{\infty}$ *with mean zero and variance* σ^2 *is called a fractional Gaussian noise with Hurst parameter* $H \in [0.5, 1)$ *if the autocorrelation satisfies*

$$R(k) = E[X_i X_{i+k}] = \frac{\sigma^2}{2}(|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}). \tag{10.4}$$

Fractional Gaussian noise is the stationary increment process of fractional Brownian motion, a continuous-time self-similar process. For $H = 0.5$, fractional Gaussian noise reduces to Gaussian white noise. For $0.5 < H < 1$, we have the following lemma.

Lemma 10.4 *Let* $R(k)$ *and* $S(\lambda)$ *be the autocorrelation function and the spectral density of a fractional Gaussian noise with Hurst parameter* $0.5 < H < 1$. *Then*

$$\begin{aligned} R(k) &\sim \sigma^2 H(2H-1)k^{2(H-1)}, \quad && as \quad k \to \infty, \\ S(\lambda) &\sim \sigma^2 C_H |\lambda|^{1-2H}, && as \quad |\lambda| \to 0, \end{aligned} \tag{10.5}$$

where $f(x) \sim g(x)$ *denotes* $\lim_{x\to\infty/0} f(x)/g(x) = 1$, *and* C_H *is a constant for fixed* H, *independent of* λ.

The above lemma tells us both $R(k)$ as $k \to \infty$ and $S(\lambda)$ as $|\lambda| \to 0$ behave like *power laws*, whose exponents are determined by the Hurst parameter H. The higher the H, the slower the decay of the autocorrelation $R(k)$ as $k \to \infty$. In fact, with $0.5 < H < 1$, $\sum_{k=-\infty}^{\infty} R(k) = \infty$, in which case we say that the random process $\boldsymbol{X}$ has *long-range dependence*, a phenomenon referred to by Mandelbrot as the "Joseph Effect." [47] Accordingly, the spectral density $S(\lambda)$ diverges at the origin.

Is fractional Gaussian noise a self-similar process? The answer is yes based on the following lemma.

Lemma 10.5 *Fractional Gaussian noise is the only Gaussian fixed point of the renormalization group, with the critical exponent equal to the Hurst parameter* H.

Another set of fixed-points of the renormalization group involve sequences of i.i.d. random variables with α-stable distributions, which will be discussed in the next section.

10.2 α-Stable Distributions

Definition 10.6 *A random variable X is said to have a* **stable distribution** *if for any $n \in \mathbb{N}$, there exist $C_n \in \mathbb{R}^+$ and $D_n \in \mathbb{R}$ such that*

$$X_1 + X_2 + \cdots + X_n \overset{d}{=} C_n X + D_n, \tag{10.6}$$

where $X_1, X_2, \ldots, X_n$ are independent copies of X. Moreover, X is called **strictly stable** *if $D_n = 0$. For any stable random variable, there is a number $\alpha \in (0, 2]$ such that $C_n = n^{\frac{1}{\alpha}}$. The number α is called the index of stability or characteristic exponent. A stable random variable X with index α is called α-stable.*

Example 10.7 *The Gaussian distribution $N(\mu, \sigma^2)$ is an α-stable distribution with $\alpha = 2$. The Cauchy distribution with the density function $\frac{\sigma}{\pi((x-\mu)^2+\sigma^2)}$ is an α-stable distribution with $\alpha = 1$.*

Lemma 10.8 *The sequence $\boldsymbol{X} = \{X_k\}_{k=1}^{\infty}$ of i.i.d. strictly α-stable random variables is a fixed point of the renormalization group $\{T_n\}_{n=1}^{\infty}$ with critical exponent $H = \frac{1}{\alpha}$.*

As shown in the previous section, if nontrivial correlations between X_k are allowed, we obtain another set of fixed points of $\{T_n\}_{n=1}^{\infty}$ which are fractional Gaussian noises with Hurst parameter $H \in [0.5, 1)$. The intersection of these two cases is Gaussian white noise with $\alpha = 2$ and $H = \frac{1}{2}$.

We have seen that both Gaussian ($\alpha = 2$) and non-Gaussian ($\alpha < 2$) stable distributions can produce self-similarity or long-range dependence in random processes. The follow lemmas explain the differences between these two distributions.

Lemma 10.9 *Let X be an α-stable random variable. Then as $x \to \infty$,*

$$\begin{cases} P[|X| > x] \sim \frac{1}{\sqrt{\pi}\sigma x} e^{-\frac{x^2}{4\sigma^2}}, & \text{for} \quad \alpha = 2; \\ P[|X| > x] \sim A_\alpha x^{-\alpha}, & \text{for} \quad 0 < \alpha < 2, \end{cases} \tag{10.7}$$

where $A_\alpha > 0$ is a constant for fixed α, independent of x.

The above lemma says the tail of an α-stable distribution with $0 < \alpha < 2$ behaves like a power law ($x^{-\alpha}$), which indicates much higher variability than the Gaussian case with $\alpha = 2$.

Lemma 10.10 *Let X be an α-stable random variable with $0 < \alpha < 2$. Then*

$$\begin{cases} E|X|^p < \infty, & \text{for any} \quad 0 < p < \alpha; \\ E|X|^p = \infty, & \text{for any} \quad p \geq \alpha. \end{cases}$$

A direct result of the above lemma is that an α-stable random variable has infinite variance when $0 < \alpha < 2$, and infinite mean when $0 < \alpha \leq 1$, which again is in sharp contrast to the Gaussian case where both mean and variance are finite.

Stable distributions can be generated through the central limit theorem. In fact, stable distributions are the only distributions that can be obtained as limits of normalized sums of i.i.d. random variables, as stated in the following theorem.

Theorem 10.11 *A random variable X has a stable distribution if and only if there is a sequence of i.i.d. random variables $\{Y_k\}_{k=1}^{\infty}$, $d_n > 0$, and $a_n \in \mathbb{R}$, for all $n \geq 1$, such that*

$$\frac{Y_1 + Y_2 + \cdots + Y_n}{d_n} + a_n \overset{d}{\Rightarrow} X, \tag{10.8}$$

where $\overset{d}{\Rightarrow}$ denotes convergence in distribution.

The special case of the limit being a Gaussian distribution ($\alpha = 2$) occurs when the Y_k have finite variance, which is also known as the ordinary central limit theorem.

When Y_k have infinite variance, their normalized sums converge to an α-stable distribution with $0 < \alpha < 2$, which, as we have seen, exhibits a power law tail, or high variability. The distributions with infinite variance are discussed in the next section.

10.3 Heavy-Tailed Distributions

We use the term *heavy-tailed* distributions for the general class of distributions with hyperbolic (power law) tails, i.e.,

$$P[X > x] \sim cx^{-\alpha}, \qquad as \quad x \to \infty, \quad 0 < \alpha < 2, \tag{10.9}$$

where $c > 0$ is a constant independent of x, regardless of the behavior of the distributions at small scales. This kind of distributions have infinite variance, a phenomenon that Mandelbrot referred to as the "Noah Effect" [46]. In practice, it means that the corresponding random variable can take on extremely large values with nonnegligible probability.

The best way to visualize a heavy-tailed distribution is to plot its complementary cumulative distribution on a log-log scale and look for approximately linear behavior in the upper tail. The slope of the straight line $-\alpha$ measures the "heaviness" of the tail, or the intensity of the "Noah Effect."

As we have seen in the previous section, α-stable distributions with $0 < \alpha < 2$ are heavy-tailed distributions. The converse is not necessarily true. One counter example is the *Pareto* distribution which is hyperbolic over its entire range. Its probability density function (PDF) is

$$p(x) = \alpha k^{\alpha} x^{-\alpha-1}, \qquad \alpha, k > 0, \ \ x \geq k, \tag{10.10}$$

and its complementary cumulative distribution function (CCDF) is given by

$$P[X > x] = (\frac{x}{k})^{-\alpha}, \tag{10.11}$$

where k is the location parameter corresponding to the minimum value for X. The above distribution is heavy-tailed when $\alpha < 2$.

A well-known result about the Pareto distribution is that it is the only distribution that satisfies the following *scaling property* [46, 4]:

$$P[X > x | X > x_0] = (\frac{x}{x_0})^{-\alpha}, \qquad x > x_0. \tag{10.12}$$

The above relation means that the distribution conditioned on $X > x_0$ is also a Pareto distribution with the same index α and a different location parameter x_0. For general heavy-tailed distributions, (10.12) is only satisfied asymptotically, as x and x_0 become large enough. This appealing property guarantees that the distribution is robust with respect to truncation from below.

Although heavy-tailed distributions may not be α-stable in general, properly normalized sums of i.i.d. heavy-tailed distributions converge to α-stable distributions with $0 < \alpha < 2$ through the central limit theorem. The latter, can in turn, give rise to random processes that are long-range dependent, or bursty over a wide range of time scales. Hence, possible links exist between heavy-tailed distributions and self-similar network traffic through α-stable distributions with infinite variance. Connections can also be made between fractional Gaussian noise traffic that is also self-similar and heavy-tailed distributions at the source level, as has been observed through empirical studies, which will be reviewed in the next two sections.

10.4 Self-Similarity Through High-Variability

In the previous sections, we discussed random processes that are self-similar or long-range dependent (the "Joseph Effect") and heavy-tailed distributions with high variability or infinite variance (the "Noah Effect"). The former has been observed in empirical studies of a large collection of Internet traffic measurements. The latter can be found in a rich body of literature on networks and computer systems.

In the search for possible physical causes of self-similar Internet traffic, an interesting connection between the above two phenomena was established by Willinger and Taqqu et al. in [73]. Based on a model originally introduced by Mandelbrot in [45], this paper claimed that the superposition of a large number of ON/OFF sources (also known as "packet trains" [36]) whose ON-periods ("train lengths") and OFF-periods ("intertrain distances") exhibit heavy tails produces aggregate network traffic that exhibits self-similarity. In fact, as both the number of sources and the block aggregation size become large enough, the cumulative traffic approaches a fractional Gaussian noise, the only stationary Gaussian process that is self-similar. Moreover, a simple relation exists between the exponent of the heavy tail ($1 < \alpha < 2$) and the Hurst parameter ($0.5 < H < 1$) of the self-similar traffic, that is

$$H = \frac{3-\alpha}{2}. \tag{10.13}$$

Generalizations of this result to more realistic settings were discussed in [68] and a rigorous proof was given. Validated by the Ethernet LAN traffic data collected at the source level, the above theory provided a plausible and simple explanation for the occurrence of self-similarity in measured network traffic in terms of the heavy-tailed nature of the traffic generated by individual sources or source-destination pairs. In contrast, traditional ON/OFF source models typically assume exponential or geometric distributions (or more generally, distributions with finite variance) for their ON- and OFF-periods. It was only in recent years when people discovered the discrepancy between the aggregate traffic generated by multiplexing a large number of these sources and the real traffic measured from various working networks. Therefore, the recognition of *heavy-tailed* distributions is indeed the essential point of departure from traditional to self-similar traffic modeling.

10.5 Heavy Tails in WWW File Transfers

Empirical studies on the subset of Internet traffic due to WWW transfers have also found evidence indicating self-similarity. One of the earliest works was presented by

Crovella et al. in [19], where a simple explanation for the self-similarity of WWW traffic was proposed, using the mechanism discussed in the previous section. In fact, the aggregate traffic can be viewed as the superposition of many ON/OFF processes, whose ON-periods correspond to transmission times of individual Web files and OFF-periods correspond to quite times in between. Measurement data from 130,140 such file transfers to 591 users on 37 machines at Boston University in early 1995 showed that the distribution of the transmission times of individual files (ON-periods) has an upper tail which declines like a power law with exponent close to 1, demonstrating extremely high variability. Moreover, the distribution of the sizes of files transferred is also heavy-tailed with a similar exponent, which is not surprising since the transmission time of a document can be roughly modeled as the sum of a cost proportional to the document size plus a fixed overhead, the latter of which can be neglected for large documents. Meanwhile, the quite times (OFF-periods) in the same data, mainly due to user think time, also exhibited heavy-tailed behavior, but with a higher exponent $\alpha \approx 1.5$. Based on the result in [68] which claimed that between the distributions of the ON- and OFF- periods, the one with the heavier tail actually determines the degree of self-similarity in aggregate traffic, the authors in [19] concluded that the distribution of Web file transfers is more likely to be responsible for the observed level of traffic self-similarity. Therefore, the focus of this thesis is to develop a series of analytical and simulation models which provide plausible explanations for the origin of heavy-tailed distributions of Web file transfers, which in turn, may be the cause for self-similar network traffic. In addition, we neglect the difference between the file sizes and the corresponding transmission times to simplify the models. The term "Web file transfers" is used in general to refer to the sizes of files transferred during Web sessions.

In Figure 10.1 the data set for the WWW file transfers mentioned above is displayed on a log-log scale (base 10), along with a data set for codewords from data compression of a file using standard methods, which we will explain in the next chapter. A line of slope $\alpha = -1$ is shown for comparison. Each data set consists of (l_i, P_i) pairs of individual events with sizes ordered as $l_i \geq l_{i+1}$ and the cumulative frequencies

$P_i = P[l \geq l_i]$. l_i can be thought of as the loss or cost proportional to the sizes of files (WWW) or lengths of codewords (DC) to be transmitted on the Internet. In sharp contrast to the data from data compression, which has an exponential distribution ($P_i \propto e^{-\lambda l_i}$), the WWW data exhibits a heavy-tailed distribution ($P_i \propto l_i^{-\alpha}$, $\alpha \approx 1$). Since the Web workloads and user behavior have changed significantly since early 1995, a new set of data was collected in 1998 in the same computing environment with 306 users on 29 machines. Again the distribution of the file transfers displays a heavy tail, although the exponent for the power law increased from approximately 1 in 1995 to 1.4 in 1998 [9]. These two data sets will be referred to as "the 1995 data" and "the 1998 data", respectively, in later chapters.

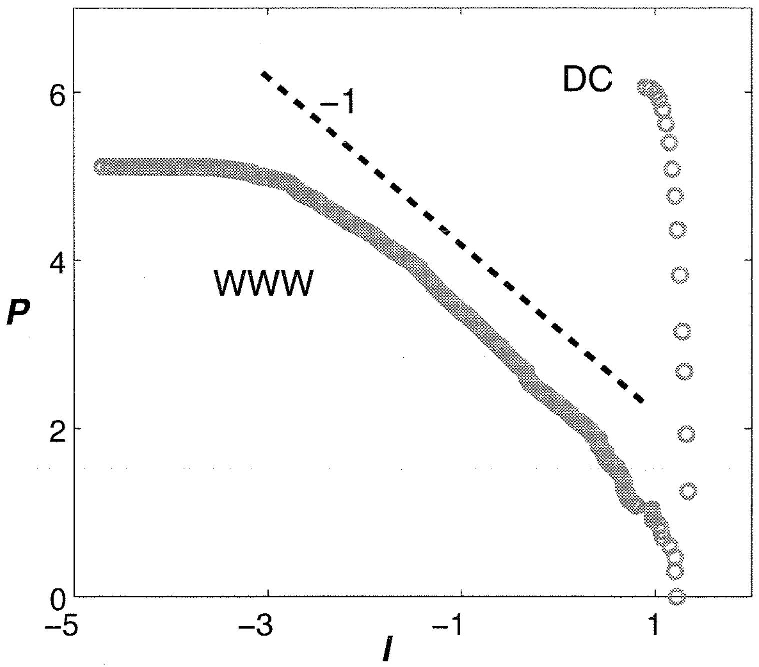

Figure 10.1: Cumulative distributions P_i vs. l_i of Web file transfers (in megabytes) and codewords from data compression.

It is worth pointing out that all the "cumulative distribution" plots in this thesis are complementary cumulative distributions on a log-log scale (base 10) for the convenience of studying heavy tails. The y-axis can be cumulative probabilities or cumulative frequencies, in the latter case $\max_i P_i > 1$, like in Figure 10.1.

Chapter 11 Generalized Source Coding

Standard source coding can be generalized using an abstract model, called the PLR model, where resources are allocated to limit average loss, e.g., the average file download time, subject to uncertainties in file content (DC) and file access (WWW), modeled probabilistically. First the PLR setup will be described abstractly, following [25], and then it is specialized to the data compression and WWW cases. For more details, implications for more general complex systems, and comparison with self-organized criticality, see [25].

For a set of abstract events with index i, $1 \leq i \leq N$, whose probabilities of occurrence are p_i, we assume there is a relationship $l_i = f(r_i)$, which describes how the allocation of resources r_i limits the sizes l_i of events. The only coupling occurs through an overall constraint on the resources available: $\sum r_i \leq R$, which only coarsely accounts for the connectivity and spatial structure of real design problems. A natural design objective in data compression and WWW is to minimize the following expected cost

$$J = \{\sum p_i l_i \mid l_i = f(r_i),\ \sum r_i \leq R\}. \tag{11.1}$$

11.1 Data Compression

While our results are completely standard and well-known in the data compression case, we will briefly review them anyway, as our notation is nonstandard to allow for generalizations to the WWW case, and otherwise could lead to some confusion. A message is a vector $\mathbf{X}$, composed of a sequence of symbols $\mathbf{X} = X_1 X_2 ....$ The X_k are independent, identically distributed random variables, chosen from N source symbols, which occur with probabilities p_i, $\sum_i p_i = 1$, and $1 \leq i \leq N$. An optimized code, consisting of codewords of length l_i, minimizes the expected length of transmission J

given in equation (11.1). For data compression the resource limitation $\sum_i 2^{-l_i} \leq 1$ is just Kraft's inequality [18], which is equivalent to unique decodability. In the PLR formulation, this becomes $l_i = f(r_i) = -\log_2(r_i)$, and $\sum r_i = R \equiv 1$. The optimal solution is $r_i = p_i$, $l_i = -\log_2(p_i)$, and the optimal cost is the Shannon-Kolmogorov entropy $J_0 = -\sum p_i \log_2(p_i)$ [18].

In the context of more general design problems, what is special about data compression is that $p_i = 2^{-l_i}$ does not have heavy tails. "Small" resource allocations are associated with long codewords, but since the number of codewords grows *exponentially* with length, rare symbols have codewords that grow in size only logarithmically. The data in Figure 10.1 uses standard Huffman coding [18] to compress the postscript file of [14]. The comparison between the data and the prediction from the PLR model will be shown in Section 11.4.

11.2 Generalized Source Coding

A more general resource versus loss function $l_i = f_\beta(r_i)$ of the following form

$$f_\beta(r_i) = \begin{cases} -\log(r_i), & \beta = 0; \\ \frac{c}{\beta}({r_i}^{-\beta} - 1), & \beta > 0 \end{cases} \tag{11.2}$$

allows treatment of more general design problems. This choice reduces to data compression ($\beta = 0$, $c = 1$), and keeps $f_\beta(1) = 0$ and $f_\beta{}'(r_i) = -c{r_i}^{-\beta-1}$, $\forall \beta \geq 0$. Note that these conditions uniquely determine $f_\beta(r_i)$ to within the constant c, which reflects a choice of units in the event sizes. The $f_\beta(1) = 0$ normalizes the loss/resource so that devoting a full unit resource eliminates loss. The choice of R reflects the total resources and the units in which resources are measured. The key quantity is the exponent β characterizing the relationship for large l_i and small r_i. It is easy to show using Lagrange multipliers that the optimal solution is

$$r_i = Rp_i^{\frac{1}{1+\beta}} \left(\sum_j p_j^{\frac{1}{1+\beta}} \right)^{-1}, \tag{11.3}$$

or

$$l_i = \begin{cases} -\log(Rp_i) + \log(\sum_j p_j), & \beta = 0; \\ \frac{c}{\beta}\left[\left(Rp_i^{\frac{1}{1+\beta}}\right)^{-\beta}\left(\sum_j p_j^{\frac{1}{1+\beta}}\right)^{\beta} - 1\right], & \beta > 0. \end{cases} \tag{11.4}$$

And the optimal cost is given by

$$J_\beta = \begin{cases} -\sum_i p_i \log(Rp_i) + (\sum_i p_i)\log(\sum_i p_i), & \beta = 0; \\ \frac{c}{\beta}\left[R^{-\beta}\left(\sum_i p_i^{\frac{1}{1+\beta}}\right)^{1+\beta} - \sum_i p_i\right], & \beta > 0. \end{cases} \tag{11.5}$$

In the special case when $R = c = 1$ and $\sum_i p_i = 1$, the above cost function becomes

$$J_\beta = \begin{cases} -\sum_i p_i \log(p_i), & \beta = 0; \\ \frac{1}{\beta}\left[\left(\sum_i p_i^{\frac{1}{1+\beta}}\right)^{1+\beta} - 1\right], & \beta > 0 \end{cases} \tag{11.6}$$

which reduces to the Shannon-Kolmogorov entropy for $\beta = 0$.

Without loss of generality we assume $r_i < 1$, since events with $r_i = 1$ ($l_i = 0$) do not contribute to the cost and can be eliminated, reducing R by 1. Inverting (11.4) yields the (noncumulative) probabilities of events of size l_i

$$p_i(l_i) = c_1(l_i + c_2)^{-(1+1/\beta)} \tag{11.7}$$

for $\beta > 0$, where c_1 and c_2 are unit-dependent constants depending on c and R.

Just as Shannon theory represents an idealized approximation to realistic data compression problems, applications of the PLR model to WWW represent very coarse descriptions of the underlying phenomena. The PLR WWW model is even more idealized than data compression, so we will later explore more complete WWW models. Nonetheless, in each case there are natural interpretations of the fundamental PLR quantities.

11.3 Application to WWW

In WWW models, we associate design with subdivision of large documents into files. The l_i are file lengths, and the p_i are determined by user interest in the files. The cost $J = \sum p_i l_i$ is the average delay a user experiences in downloading files because the transmission time of each reasonably large file is approximately proportional to the file size. For really small files this may not be true due to the nonnegligible fixed overhead for the transmission, so there is a small scale cutoff on l_i. The resources r_i divide the document into files, where large r_i correspond to small files and small r_i correspond to large files. The tradeoff between the lower cost of shorter files, and the need for efficient Web management and user navigability which favors larger files leads to a resource constraint on the number of files.

The most subtle aspect of the PLR model is determining β to reflect the resource versus loss tradeoff. We begin by assuming that a Web site is created by taking a one-dimensional document and splitting it into files which are then connected by links. Subdividing a one dimensional document into files with zero dimensional cuts leads to an inverse relationship between the file size and the resource density, $l_i \propto {r_i}^{-1}$, so that $\beta = 1$. To go beyond this rough dimensional argument and construct a microscopic model for which the PLR formalism (and ultimately also the dimensional argument) applies exactly, we must also insure that the p_i are not changed by varying the r_i. Suppose we split the document into N regions of equal size L and assume a fixed probability p_i of a user hit occurring in the i^{th} region. We then design cuts to subdivide the i^{th} region into n_i equal files each of size l_i. Thus, the size is $l_i = n_i^{-1} L$, while the resource allocation in the i^{th} region is $r_i \propto n_i$, where proportionality reflects possibly differing units. This yields $l_i \propto r_i^{-1}$, again giving $\beta = 1$. In [25], the more general case where $\beta = d$ is considered, which reflects the natural relationship between the size of $d-1$ dimensional resources traded off against the size of the resulting d dimensional events.

11.4 Comparison with Data

The data sets for data compression and WWW shown in Figure 10.1 are replotted in Figure 11.1 for comparison with the PLR model. Each data point (l_i, P_i) shows the cumulative frequency P_i versus the event size l_i, where l_i are ordered as $l_i \geq l_{i+1}$. To generate corresponding data $(l_i, \hat{P}_i)$ for the PLR model, we choose β based on the resource vs. loss relationship for a given system ($\beta = 0, 1$ for DC and WWW, respectively). To relate the noncumulative p_i for the model to the cumulative P_i for the data, we evaluate (11.7) for each l_i in the data set. The difference approximation to $p = dP/dl$

$$p_i = (P_{i+1} - P_i)/(l_i - l_{i+1}) \tag{11.8}$$

can be inverted to approximate each P_{i+1} in terms of all the p_i and l_i, i.e.,

$$\begin{aligned} \hat{P}_{i+1} &= \sum_{j \leq i} p_j (l_j - l_{j+1}) \\ &= \sum_{j \leq i} c_1 (l_j + c_2)^{-(1+1/\beta)} (l_j - l_{j+1}). \end{aligned} \tag{11.9}$$

The constants c_1 and c_2 in (11.9) (equivalently R and c) are set by the cutoff size of the smallest events in the data set, and the total frequency of events, and thus effect the position but not the general shape of the curves. The data compression problem has only the constant c_1, since for DC $c_2 = 0$.

In Figure 11.1 the data sets $(l_i, \hat{P}_i)$, $i \geq 2$ from the PLR model (dashed line) for $\beta = 0,\ 1$ are compared directly and unambiguously with the original data (l_i, P_i) (grey circles). The agreement is excellent, especially for the WWW data. The slight discrepancy in the DC case is a consequence of finite block sizes (in this case 16 bits) and integer codeword lengths. Both are essential practical issues not captured in the idealized Shannon theory discussed in Section 11.1.

The PLR model provides some insights into the origin of power laws in HOT systems [14, 15] in general, and the special characteristic in data compression that

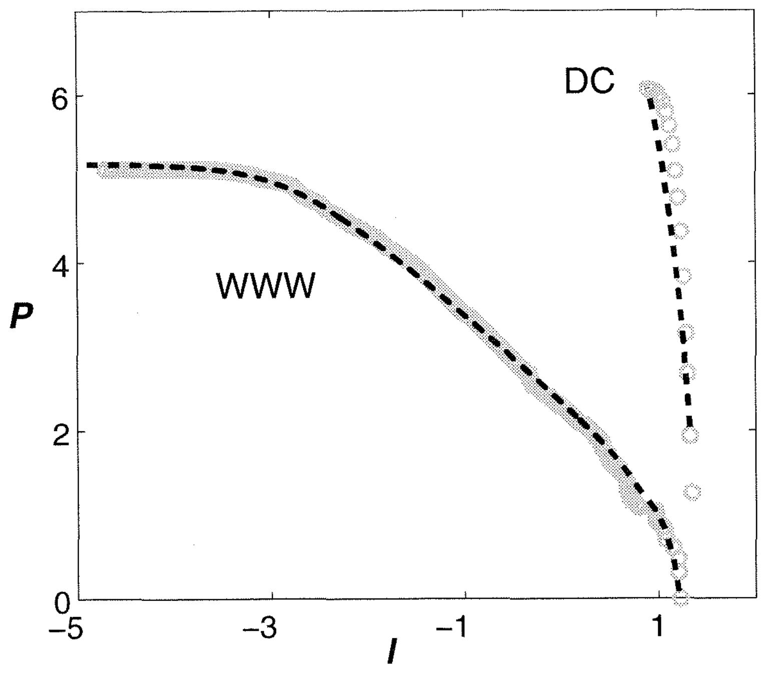

Figure 11.1: Comparison of DC and WWW data (l_i, P_i) with the results of the PLR model $(l_i, \hat{P}_i)$, $i \geq 2$.

leads to nonheavy-tailed optimal distributions. Its application to Web file transfers provides a plausible explanation for the self-similar behavior of WWW traffic, given the results discussed in the previous chapter. On the other hand, due to its simplicity, the PLR model is an extreme abstraction of the complicated Web design process in the real world. Of course, it is unlikely that individual Web sites in 1995 were optimally designed, so the agreement of model with data could be coincidental. This issue will be discussed at the end of the thesis, after we examine more realistic Web layout models and see if similar agreement still holds.

Chapter 12 A More General PLR Model and Optimal Web Layout Design

The PLR model defined in (11.1) assumes that only the loss l_i depends on the resource r_i, while the probability of each event p_i remains fixed, which allows an analytical solution to be obtained from the optimization. This constraint is rather restrictive, which may be violated in more practical Web layout design. The following PLR model

$$J = \{\sum p_i l_i \mid l_i = f(r_i),\ p_i = g(r_i),\ \sum r_i \leq R\} \tag{12.1}$$

generalizes (11.1) to the cases where the probability p_i may also be affected by the resource allocation r_i. This fairly broad setting makes the optimization problem extremely hard to solve analytically. However, this model can also be simplified by making some further assumptions. In these special cases, optimal solutions for (12.1) can be calculated either numerically or even analytically. The one-dimensional Web layout model introduced in this chapter is an example of this specialization. More complicated and realistic models of Web site design can be built on top of this, in which case global solutions to (12.1) are not guaranteed to be found. Simulations of the optimization procedure using heuristic algorithms on graph-based Web layout models will be presented in the next chapter.

The model proposed in this chapter is motivated by the PLR model for WWW reviewed in the previous chapter, but is actually closer in detail to the models studied in [14]. By focusing on the simplest possible setting—a one-dimensional Web layout problem, the model allows for analytical treatment. At the same time, it captures some additional features of a real Web site beyond what is in the PLR model in [25]. In particular, this new model includes a hierarchical structure for the Web site and a more reasonable model of user navigation behavior. We begin with the case of building a Web site out of a single document, then discuss the more complex scenario with multiple documents.

12.1 One-Dimensional Web Layout Model

The one-dimensional Web layout model considers the early stage of Web site design when people basically just put preexisting documents onto the Web. When such a document is fairly big, it is unlikely that all users are interested in every portion of the document. Then it is necessary to split the document into smaller files and connect them in a simple way so that the average latency a user experiences while downloading the relevant information is minimized.

A real random variable X with the sample space $[0, L]$ can be used to model user interest in various portion of the document, where L is to total length of the document with certain units, e.g., bytes. The PDF $p(x)$ indicates the popularity of the information at location x. The specific functional form of $p(x)$ may vary with each individual document. Here we would make a reasonably generic assumption that the document is organized in such a way that more popular information is placed before less popular information, which means $p(x)$ is strictly decreasing with x, as shown in Figure 12.1.

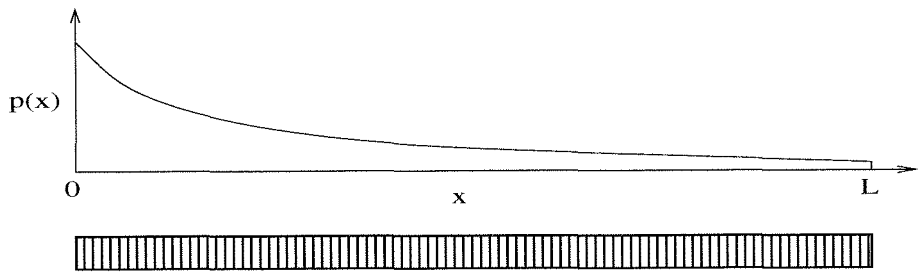

Figure 12.1: Document of size L.

Now divide the document into N files, with the locations of the cuts being $c_1, c_2, \ldots, c_{N-1}$. Denote the size of each file as l_i and the average popularity of each file as p_i. Then for any i, we have

$$\begin{cases} l_i = c_i - c_{i-1}, \\ p_i = \int_{c_{i-1}}^{c_i} p(x)dx, \end{cases} \tag{12.2}$$

where $c_0 = 0$, $c_N = L$. This division is illustrated in Figure 12.2. Note that $\sum_{i=1}^{N} l_i = L$, and $\sum_{i=1}^{N} p_i = 1$.

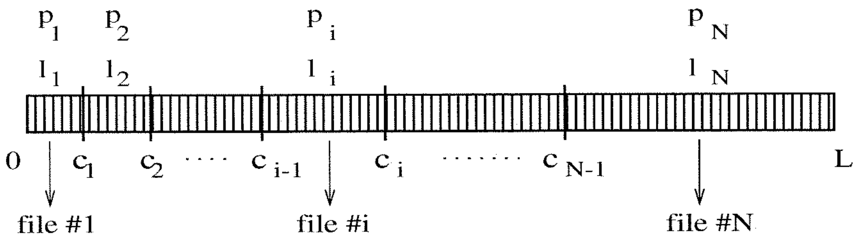

Figure 12.2: Dividing the document.

A one-dimensional Web site can be made by sequentially linking the above N files, so that each file corresponds to a Web page, as shown in Figure 12.3. The design problem is to minimize the average download time of the Web site by optimally dividing the original document, which can be modeled using the general framework in (12.1), but in a simplified setting. At one extreme, it is always desirable to divide the document into as many small files as possible to lower the cost, which, however, sacrifices the ease of manageability and navigability of the Web site. It also causes the Web server to process more requests for the same amount of information. Moreover, when a file is sufficiently small, the overhead associated with the transmission dominates the time required to transfer the file, which diminishes the advantage of making small files. So the number of files N is assumed to be fixed as a constraint on limited resources, like the constraint $\sum_i r_i \leq R$ in the PLR model. When N is not too large, we assume all the resulting files are big enough that the download time of each file can be approximated by the file size, with proper arrangement of the units. To take into account the navigation behavior of users, we also assume that any user who visits the Web site has to start from the first page, and go through pages in ascending order. This means a user interested in page $\#i$ must access pages $\#1$ through $\#i$.

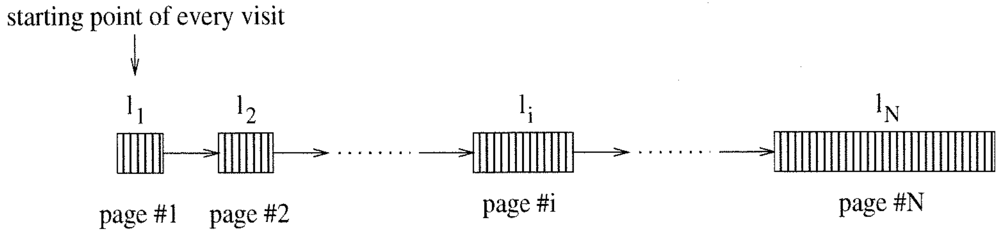

Figure 12.3: A one-dimensional Web site.

In summary, page $\#i$ will be targeted with probability p_i from each user's visit. Then all the pages from $\#1$ through $\#i$ are transferred, with the total download time equal to the total size of the files, which is $\sum_{j=1}^{i} l_j = c_i$. Thus, minimizing the average download means minimizing the following cost function

$$J_1 = \sum_{i=1}^{N} p_i c_i \tag{12.3}$$

with respect to the cut locations $c_1, \ldots, c_{N-1}$.

An equivalent way to formulate the problem is the following. Denote the hit rate of file $\#i$ among all file transfers as p_i^t, then $p_i^t = \sum_{j=i}^{N} p_i$, because file $\#i$ will be transferred whenever a user is interested in file $\#j$ with $j \geq i$. Note that p_i^t are not probabilities since $\sum_{i=1}^{N} p_i \neq 1$. However, we can say that for each file transfer, the download time is l_i with frequency p_i^t. Hence, the average download time becomes

$$J_2 = \sum_{i=1}^{N} p_i^t l_i \tag{12.4}$$

which can be minimized with respect to the sizes of individual files $l_1, \ldots, l_N$.

It is easy to show that $J_1 = J_2$. So the above two optimization problems are exactly the same. To derive the optimality condition, let $\frac{\partial J_1}{\partial c_i} = 0$, $i = 1, \ldots, N-1$. Then the following recursion relations hold:

$$p(c_i) l_{i+1} = p_i, \qquad i = 1, \ldots, N-1. \tag{12.5}$$

Since $p(x)$ is a strictly decreasing function, it is easy to verify that $l_i < l_{i+1}$ based on (12.2), which means the optimal file sizes will strictly increase with the index. Unfortunately the exact values of the optimal l_i can only be solved numerically for general $p(x)$. For analysis purposes we need to make further assumptions on $p(x)$, for instance, we can assume $p(x)$ belongs to certain classes of distributions that are plausible for modeling user interest and have distinct characteristics. One such class is the exponential distribution. This occurs when users navigate through the document with a memoryless process, i.e., $P[X > x_0 + x | X > x_0] = P[X > x]$. It means that the probability of a user going through a segment of length x is independent of the starting point, and the exponential function is the only continuous PDF that meets this criterion. It will be interesting to see what the distribution of file transfers looks like on an optimized Web site. So assume that

$$p(x) = ke^{-\lambda x} \tag{12.6}$$

where $0 \leq x \leq L$, and $k = \frac{\lambda}{1-e^{-\lambda L}}$. Let L be big enough so that $e^{-\lambda L} \ll 1$, and $k \doteq \lambda$. By substituting (12.6) into (12.5), we get the recursion equations for the optimal file lengths:

$$l_{i+1} = \frac{e^{\lambda l_i} - 1}{\lambda}, \qquad i = 1, \ldots, N-1. \tag{12.7}$$

The above relation indicates that the sizes of the files grow exponentially towards the end of the document. It is predictable that the last file in the sequence will be significantly larger than the first file. Obviously it is beneficial to group less popular information into big files while keeping the sizes of popular files small. The above equations can be solved numerically with the constraint $\sum_{i=1}^{N} l_i = L$. With the optimal solutions for l_i, we can compute $c_i = \sum_{j=1}^{i} l_j$, and furthermore, $p_i^t = \int_{c_{i-1}}^{L} p(x)dx = e^{-\lambda c_{i-1}} - e^{-\lambda L}$. Now normalize p_i^t such that $\sum_{i=1}^{N} p_i^t = 1$. If we consider the size of the file transferred as a random variable l, then the sample space is $\{l_1, l_2, \ldots, l_N\}$, and p_i^t is the probability mass function(PMF) of l over each sample. The cumulative probability $P_i^t = P[l \geq l_i]$ vs. the size l_i is shown in Figure 12.4, with

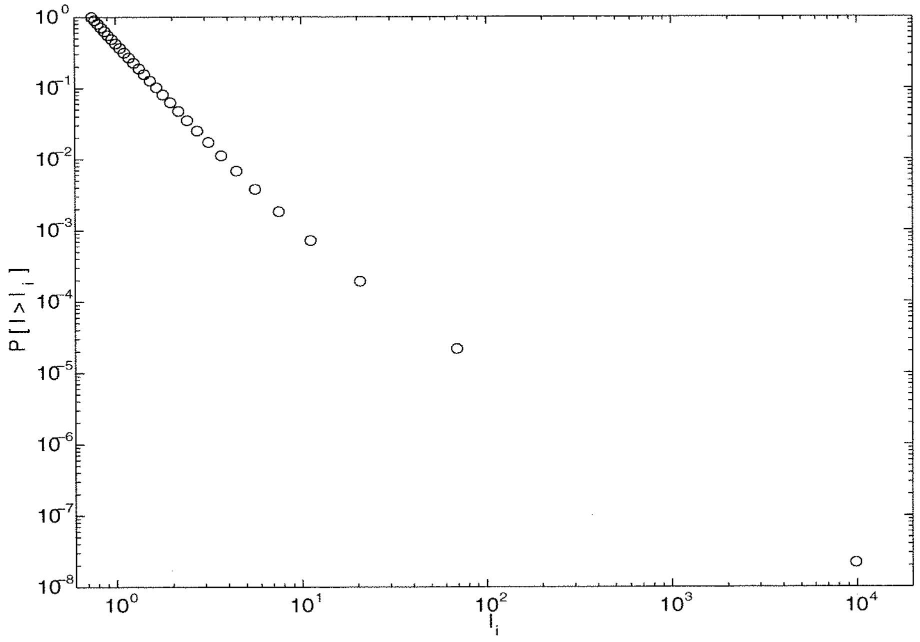

Figure 12.4: P_i^t vs. l_i for Web file transfers. ($p(x)$ is exponential.)

$L = 10^4$, $N = 20$ and $\lambda = 0.1$.

The cumulative distribution of transferred file sizes displays an asymptotic behavior that is consistent with a heavy-tailed distribution. The slope of the curve in a log-log scale increases as the file size l_i grows bigger, and approaches -1 asymptotically as l_i becomes sufficiently large. This characteristic of the distribution is quite robust with respect to different parameter values N and λ as long as L is large enough.

To prove this asymptotic behavior analytically, we need to consider the limit case where $L = \infty$. Then the probability of transferring file $\#i$ becomes $p_i^t = e^{-\lambda c_{i-1}}$. It follows that as $l_i \to \infty$,

$$\frac{\log p_{i+1}^t - \log p_i^t}{\log l_{i+1} - \log l_i} = \frac{\log e^{-\lambda c_i} - \log e^{-\lambda c_{i-1}}}{\log \frac{e^{\lambda l_i - 1}}{\lambda} - \log l_i} = \frac{-\lambda l_i}{\log(e^{\lambda l_i} - 1) - \log \lambda l_i} \sim \frac{-\lambda l_i}{\lambda l_i - \log \lambda l_i} \sim -1,$$

where $f(x) \sim g(x)$ denotes $\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1$, as defined in Chapter 10. The above argument shows that the slope of p_i^t vs. l_i in a log-log plot approaches -1 as l_i approaches infinity, which means $p_i^t \sim c l_i^{-1}$ where c is a constant. So the probability of transferred file size has a heavy tail with exponent 1. In reality the file sizes are always finite. But we can view each l_i as being infinite when $l_i \gg l_1$. Even for medium sized l_i, the slope $\frac{-\lambda l_i}{\lambda l_i - \log \lambda l_i}$ is fairly close to -1. And because $l_i \ll l_{i+1}$, $p_i^t \gg p_{i+1}^t$, the cumulative distribution of transferred file sizes P_i^t has exactly the same asymptotic behavior, which is verified by the numerical result demonstrated in Figure 12.4.

Some people may argue that user navigation of a Web site is not necessarily memoryless, which means $p(x)$ may not be exponential. So two other commonly used distribution functions were also tested: the Gaussian distribution ($p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$) and the Cauchy distribution ($p(x) = \frac{\lambda/\pi}{1+(\lambda x)^2}$). These two were chosen because they are representative of distributions with different types of tails. Both are α-stable distributions. The Gaussian distribution has $\alpha = 2$ implying a finite variance, while the Cauchy distribution has infinite variance and infinite mean with $\alpha = 1$. So the Cauchy distribution is itself heavy-tailed, while the Gaussian distribution is far from being heavy-tailed. With these distributions, a simple recursion relation between l_i as the one in (12.7) is not available. However, similar asymptotic results on the tail of the probability of file transfers can be shown. See Appendix A for the proof of these results. Here we solve the optimization numerically using equation (12.5).

Again, we set $L = 10^4$, and $N = 20$. In both cases, the resulting cumulative distributions P_i^t vs. l_i display heavy tails, as shown in Figure 12.5. The Gaussian case is similar to the exponential one, where most of the resulting l_i cluster around size 1 and a few bigger ones spread out quickly towards L. Again the distribution is not exactly a power law, but the slope of the curve approaches -1.4 in the region of large l_i. With the Cauchy distribution the resulting l_i are evenly distributed on a logarithmic scale and their cumulative probabilities can be very well approximated by a power law with exponent 1.

To summarize, the above results suggest that the optimal layout of a one-dimensional Web site with the objective of minimizing the average download time subject to a constraint on the number of files leads to the heavy-tailed distribution of transferred file sizes with exponent $\alpha \approx 1$. This confirms the observation from the PLR model in Chapter 11, and is consistent with the 1995 data. The result is quite robust with

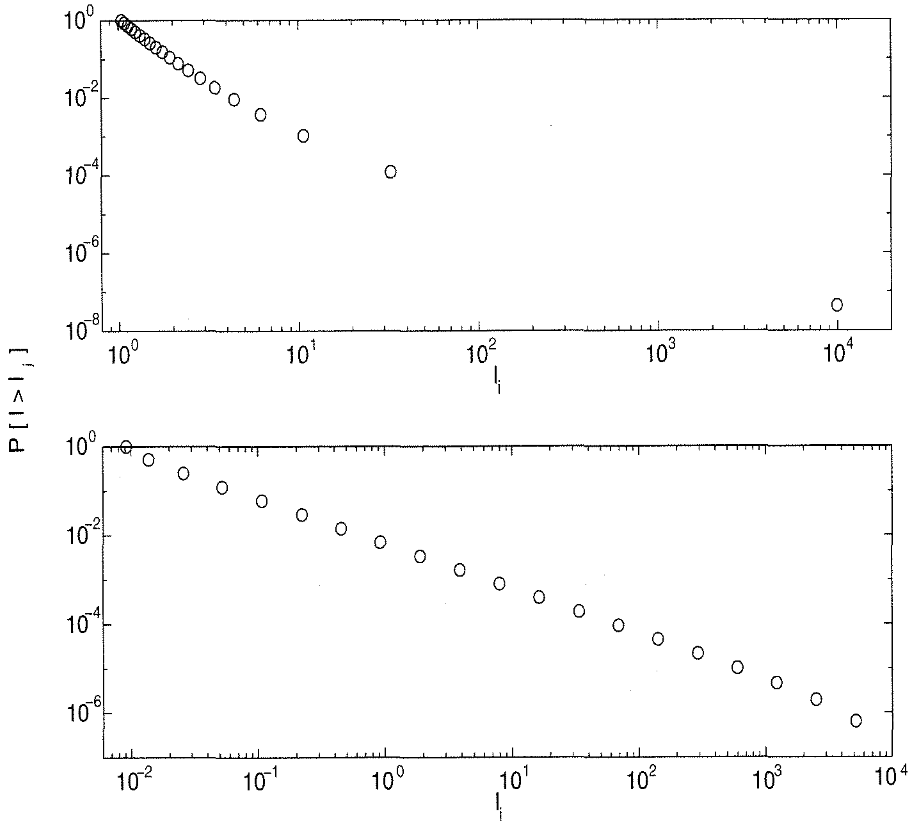

Figure 12.5: P_i^t vs. l_i for Web file transfers. (Top: $p(x)$ is Gaussian, $\mu = -30$, $\sigma = 20$; Bottom: $p(x)$ is Cauchy, $\lambda = 100$.)

respect to the specific choice of the initial $p(x)$ and whether it is heavy-tailed or not. Although the behavior of the model under distributions other than those tested above is subject to verification, we do believe that the link between the heavy-tailed distribution of Web file transfers and the optimization behind Web layout design and the tradeoff between performance objective and resource constraints is fairly generic. Of course the above study is not conclusive since the model of Web site layout and user behavior is so simplified. The rest of the thesis addresses the issue of investigating more complex models for Web layout design.

12.2 Web Sites of Multiple Documents

It is possible to extend the model in the previous section to Web sites consisting of multiple documents. Imagine such a Web site is designed by dividing each document into a one-dimensional chain in the optimal way described in the previous section. The goal is to study the statistics of the overall traffic from all the documents. Assume that each document is equally likely to be visited. And because previous study indicates that the optimal design for each document is qualitatively quite independent of the specific choice for the initial $p(x)$, for simplicity we would further assume that the popularity of the information in each document is i.i.d. with an exponential distribution $p(x) = \lambda_0 e^{-\lambda_0 x}$. The analysis of a single document has shown that the key to the resulting heavy-tailed distribution is the recursion relation (12.7) between sizes of subsequent files. Therefore, if the distribution of the smallest file l_1 in each chain is known, it can be propagated through the recursion equation and obtain the overall distribution of all the files. In general the sizes of small files do not have heavy-tailed distributions, so the distribution of l_1 is assumed to be an exponential function on an interval I_1 with parameter λ_1,[1] i.e., its PDF $f_1(l_1) = k_1 e^{-\lambda_1 l_1}$, $l_1 \in I_1$.

Let $h(x) = \frac{e^{\lambda_0 x}-1}{\lambda_0}$. Then $l_i = h(l_{i-1}) = h^{i-1}(l_1)$, for all $i > 1$, where $h^m(\cdot)$ denotes the m_{th} composite function of $h(\cdot)$. The inverse function of $h(x)$ is $g(x) = \frac{\log(\lambda_0 x+1)}{\lambda_0}$, so $l_1 = g(l_2) = g^{i-1}(l_i)$. Then the PDF of $l_i (i > 1)$ is

$$f_i(l_i) = f_1(l_1)\frac{\partial l_1}{\partial l_i} = f_1(g^{i-1}(l_i))\frac{\partial g^{i-1}(l_i)}{\partial l_i}, \tag{12.8}$$

which is a highly complicated function of l_i. However, because each l_i is an exponential function of the previous one, it is conceivable that the cumulative distributions of l_i spread out exponentially as i increases.

The size of the file transferred from this Web site can be considered as a new random variable l_m, whose distribution will be a mixture of all the distributions $f_i(l_i)$. And the mixing probability p_i^t is also a function of the particular value of l_i, which can be calculated as we described in the previous section. By the theory of mixture probability [29], the cumulative distribution $P_m(x) = P[l_m > x]$ is given by

[1]Note that λ_1 is for the distribution of l_1 while λ_0 is for the distribution of user interests in each document.

$$P_m(x) = \sum_{i=1}^{N} \int_x^\infty p_i^t(l_i) f_i(l_i) dl_i. \tag{12.9}$$

The above distribution is computed numerically and the resulting P_m vs. l_m is displayed in Figure 12.6 (dashed line). The upper tail of the distribution is very well approximated by a power law with exponent 1.1. To illustrate the cause of this heavy tail, the cumulative distributions of l_i conditioned on transferring file $\#i$ in the chain are also shown (solid lines). The plot is cut off at $x = 10^{10}$ to show to details of how the solid curves add up to give the dashed curve. Here we used $N = 10$, $\lambda_0 = 0.25$, and $\lambda_1 = 1$. The shape of the distribution is quite insensitive to the choice of N and λ_1. As λ_0 gets bigger, the distributions of $l_i (i > 1)$ spread out more so that the exponent of the power law tail gets closer to 1. If λ_0 is too small, no big files will be generated from this iteration, which is of no interest to this study.

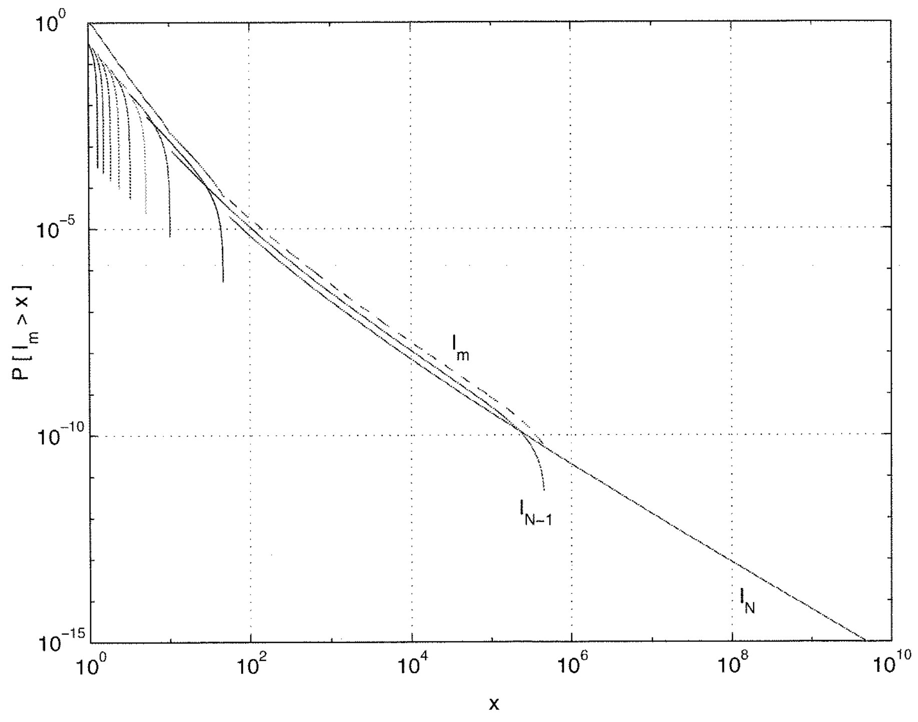

Figure 12.6: Cumulative distribution of l_m.

Chapter 13 Simulations of Graph-Based Web Layout Models

The one-dimensional Web site model is good for analytical study, but is only representative of fairly primitive Web layout design. Recently, the layout of Web sites has become much more complicated and versatile and the design process has gone far beyond just organizing a single document, which means the one-dimensional assumption is no longer suitable. The objective of this chapter is to build a series of models with increasing details of real Web layout and minimize the average download time, i.e.,

$$J = \sum_i p_i l_i \tag{13.1}$$

where l_i are the sizes of individual files on a Web site, and p_i are the corresponding probabilities that each file gets downloaded. Again constraints are posed due to limited resources. The resulting optimization problem fits into the general PLR model in (12.1). Its combinatoric nature implies that analytical solutions will typically not be available. Instead, heuristic optimization schemes have been developed and are presented in this chapter, which mimic some aspects of the tuning and refining involved in real Web site design, and suggest simple methods by which this could be made more systematic.

The next section introduces random graphs as more realistic models for general Web sites. Graph-based models have been employed by many other researchers for topologies of computer networks [71, 23] and the WWW [8, 49, 34, 70]. In [8] the Web was viewed as a large directed graph whose vertices are documents and edges are links pointing from one document to another. In [34] the graph model was applied on a single Web site, which is closer to its application in this chapter — the study of the layout of each individual Web site. The optimization of the average download time is done by heuristically reorganizing (splitting and merging) files based on statistics of user interest. Simulations of different classes of graphs including chains, trees, and incrementally generated networks are described in Section 13.2. In Section 13.3 a more sophisticated model with geometries in individual Web pages is presented as well as simulations with a slightly more involved optimization procedure.

13.1 Web Sites as Random Graphs

Consider a Web site modeled as a directed graph (V, E), where each node $V_i (i = 1, \ldots, N)$ represents a Web page and each directed edge E_{ij} represents a hyperlink pointing from page V_i to page V_j. This model still contains a fair amount of simplification from real Web sites, for instance,

1. The geometry inside each file is ignored, so each file is an abstract node in the graph;
2. Each Web page contains only a single file. All the embedded objects (e.g., images) are considered to be part of that file. Hence, no distinction will be made between a file and a Web page;
3. The effect of caching is not taken into account so that every hit on a Web page leads to a transfer of the corresponding file.

All these assumptions can be removed by adding more complexity to the model. The model described in Section 13.3 is an example of relaxing the first condition. The implication of the no caching assumption will be discussed later along with simulation results.

13.1.1 The Markov Chain Model

A discrete-time Markov chain is used to model user navigation behavior, where all the nodes in the graph are considered as possible states for the random process, and the probability of going from V_i to V_j defines the transition probability p_{ij}. Let $M = [p_{ij}]$ be the transition probability matrix, then $M^n = [p_{ij}(n)]$ gives the n-step transition probabilities. The following theorem is from the standard theory of Markov chains.

Theorem 13.1 *[43] For an irreducible, aperiodic, and positive recurrent Markov chain,*

$$\lim_{n\to\infty} p_{ij}(n) = p_j, \qquad \textit{for all } j,$$

where p_i are the unique nonnegative solutions of the following equations:

$$p_j = \sum_i p_i * p_{ij}, \tag{13.2}$$

$$\sum_i p_i = 1. \tag{13.3}$$

A Markov chain that satisfies the conditions stated in the theorem is called ergodic, which is true for the Markov chains defined in all the models in this chapter. Asymptotically, an ergodic Markov chain settles into a stationary random process with state probabilities p_i independent of the time instant n. Each p_i also corresponds to the long-term proportion of time spent on state V_i, in other words, the probability of page V_i being accessed, or downloaded, which is exactly the p_i in the cost function (13.1). Suppose the transition probability matrix M is know, then p_i can be computed by solving equations (13.2) and (13.3). More specifically, let $p = [p_1, \ldots, p_N]^T$ (column vector), we can write (13.2) in its vector form, which is

$$p' = p'M \qquad \text{or} \qquad p = M'p. \tag{13.4}$$

This means p is the right eigenvector of M' with an eigenvalue 1. For real Web sites, p_i can be directly calculated from log data with sufficiently large sample size. More advanced technology even allows the server to trace every link that a user follows, which provides information on the transition probabilities p_{ij}, or, the matrix

M, from which p_i can be obtained.

To further define the Markov process, page V_1 is set as the entry point of every user's navigation, assuming that a user starts from the front page and then proceeds to subsequent pages through hyperlinks. So after downloading page V_i, the user follows hyperlink E_{ij} to visit page V_j with probability p_{ij}. All $p_{i1} (i > 1)$ are viewed as probabilities of exiting the Web site from page V_i, which means the user either stops navigating or goes on to other Web sites. Here instead of defining another node as the exiting state, we let all users go to node V_1 when they exit because the overall statistics are an aggregation of many users' navigation behavior, and there is no way to tell the difference between user A going back to V_1 and starting over again and user B entering at V_1 and beginning to navigate.

13.1.2 Optimization through Splitting and Merging

Suppose the original Web layout has N files connected as a graph. The length of each file is l_i, and the probability of each file being downloaded is p_i. A naive way to reduce the average download time is to split high hit files and merge unpopular ones. In particular, if a file is popular and large at the same time, split it into two smaller files. Repeatedly doing this will certainly lower the cost defined in (13.1). However, the total number of files N increases by 1 with every split. Eventually the number of files becomes too large, which leaves the Web site too hard to manage and users frustrated by going through too many clicks. Therefore, as a coarse approximation of the tradeoff between the cost and the above concerns, an upper limit is posed on the total number of files, as in the one-dimensional Web layout model. To meet this constraint, every time one file gets split up, two other files that neighbor[1] each other and are rarely visited are chosen to be merged into a larger file. This is the basic idea behind the heuristics used in the optimization. Either splitting or merging changes the topology of the graph, the corresponding Markov chain, and the distribution of the file sizes. The details involved in splitting and merging depend on individual

[1]Two nodes are called "neighbors" if they are connected by hyperlinks.

models, and will be discussed in the next section on specific simulations.

13.2 Simulations on Random Graphs

13.2.1 Initialization of the Graph Model

The first step of the simulation is to create the initial topology of the graph. Two parameters are chosen to provide simple control over the key structural characteristics of a Web site: N_{node}, the number of nodes in the graph, determines the scale of the Web site under study, and N_{link}, the number of incoming links at each node, governs the degree of connectivity between the Web pages. The graph model is built in an incremental way to mimic the generation of real Web sites, where people initially put up some Web pages with hyperlinks in between, then gradually add more pages and connect them with the existing pages, and keep growing the Web site in an iterative fashion. Each node V_i has coordinates (x_i, y_i), which are generated uniformly in a 10×10 square in $\mathbb{R}^2$. This is for the convenience of modeling the logic relation between different pages. The relevance of information between pages V_i and V_j is represented by their distance in Euclidean space, the same idea used in Waxman's network model [71]. Each graph usually starts with a few nodes. Every time a new node is generated, it is connected with N_{link} of the existing nodes that are closest to it with bidirectional links. This process is repeated until all N_{node} nodes are generated and connected. Figure 13.1 shows one graph model generated by the above scheme. V_1 corresponds to the starting point of the navigation.

Notice that most nodes are linked to nearby neighbors, while there are occasionally rather long links connecting neighbors that are distant from each other on the graph. This property is similar to the small world phenomena present in the WWW, discussed in [1]. The reason is that the graph is built incrementally. So these neighbors that are far away are those generated earlier when there were only a few nodes on the graph, while all the others were generated subsequently. This indicates some intrinsic hierarchy that is present in many Web sites, where the front page connects to a small

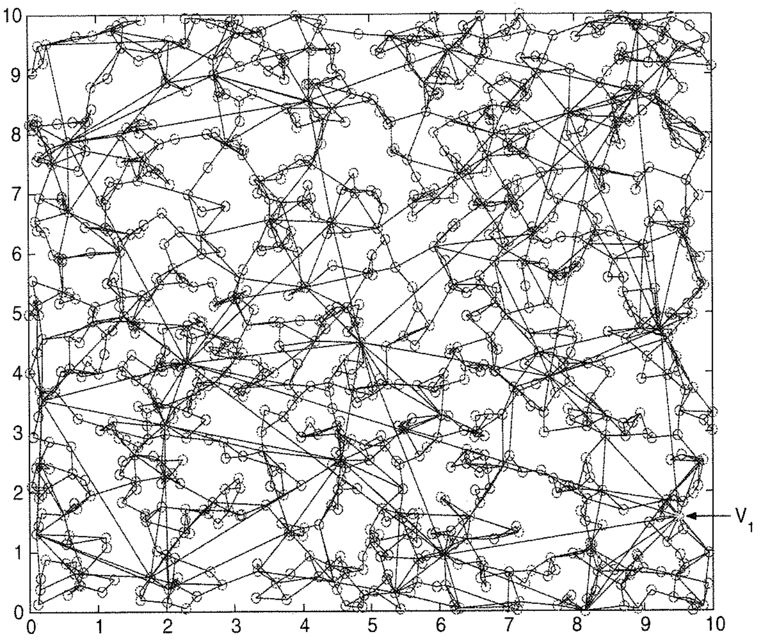

Figure 13.1: Graph model of a random Web site with $N_{node} = 1000$ and $N_{link} = 2$.

number of key pages with higher level information, each of them surrounded locally by a large number of lower level pages with more detailed information.

Real Web site layout could be fairly different from one to another. Besides the two parameters N_{node} and N_{link} chosen to control the scale and connectivity of a Web site, the randomness in the locations of the nodes and consequently the connections between them are used to provide the degree of freedom in the topology. The difference in the degree of connectivity between $N_{link} = 1$ and $N_{link} = 3$ is clearly demonstrated in Figure 13.2.

After interconnections between the Web pages are determined, the user navigation pattern is modeled by assigning the initial transition probabilities p_{ij} of the Markov chain. For each node V_i, simply assume that there is one common exiting probability p_e except for V_1, i.e., $p_{i1} = p_e (i > 1)$. The remaining probability $1 - p_e$ will be evenly distributed among all the outgoing links $E_{ij} (j > 1)$. Set $p_{ij} = 0$ if there is no link between two nodes V_i and V_j. Note that the particular Markov chains defined here do not have self-loops, i.e. $p_{ii} = 0$. After uniquely determining the whole transition probability matrix M, the download probabilities p_i can be computed

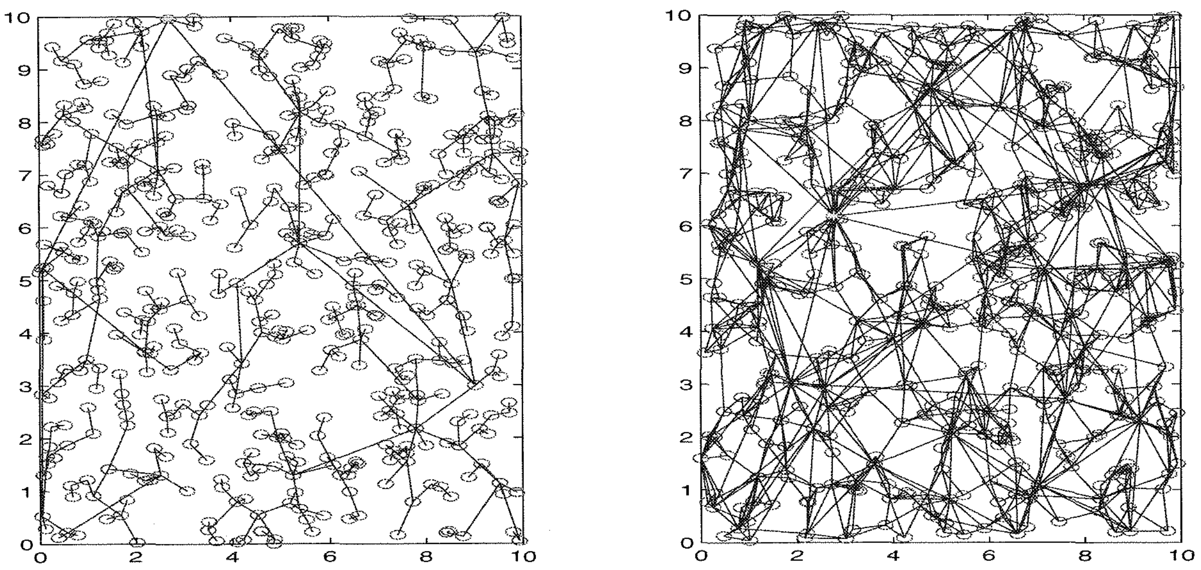

Figure 13.2: Graph models of two random Web sites with $N_{node} = 500$ and different degrees of connectivity. Left: $N_{link} = 1$; Right: $N_{link} = 3$.

through eigenvalue decomposition, or solving (13.4).

The last step of the initialization is to decide the file size l_i for each page. Three cases have been tested: the exponential distribution, the uniform distribution in $[l_{min}, l_{max}]$, and the degenerate case where $l_i = L$, for all i. In the next section, the simulation results with initial exponential distributions for l_i are shown. The results with the other two kinds of distributions are fairly similar, hence will be omitted here.

13.2.2 Heuristic Optimization and Simulation Results

The heuristic optimization algorithm goes as follows:

Splitting Pick page V_{i^*} with the highest $p_i l_i$. Cut the page in the middle to produce two new pages $V_{i'}$ and $V_{i''}$ with sizes $l_{i'} = l_{i''} = l_i/2$. Any outgoing link E_{ij} becomes $E_{i'j}$ and $E_{i''j}$ without changing the outgoing probability, i.e., $p_{i'j} = p_{i''j} = p_{ij}$. Meanwhile, any incoming link E_{ji} becomes $E_{ji'}$ and $E_{ji''}$ with half incoming probability, i.e., $p_{ji'} = p_{ji''} = p_{ji}/2$. The download probability vector p remains almost the same except that $p_{i'} = p_{i''} = p_i/2$. It is easy to prove that the updated M and p still satisfy equation (13.4).

Merging After splitting one page, find page V_{k^*} with the lowest p_k and its least popular neighbor $V_{k^{**}}$. Combine them into one page V_k with size $l_k = l_{k^*} + l_{k^{**}}$. Copy all the the outgoing links and incoming links for V_{k^*} and $V_{k^{**}}$ into the new page, then merge redundant links and combine probabilities, i.e., $p_{jk} = p_{jk^*} + p_{jk^{**}}$, and $p_{kj} = \frac{p_{k^*}}{p_{k^*}+p_{k^{**}}} p_{k^*j} + \frac{p_{k^{**}}}{p_{k^*}+p_{k^{**}}} p_{k^{**}j}$. The self-loop produced by merging should be removed and all $p_{kj}, j \neq k$ need to be adjusted appropriately so that each row of M still sums up to 1. Recompute the download probability vector p using equation (13.4).

Iteration Update $J = \sum_i p_i l_i$. Repeat the above splitting-merging procedure until the improvement of J is within a certain tolerance level or the number of iterations reaches a preset maximum.

The algorithm described above is the most naive version so that it contains the key operations and is easy to understand. The real implementation involves a fair number of variations of the above algorithm. For example, the criterion for choosing the page to split or the pages to merge can be different; while splitting, the location of the cut does not have to be in the middle, instead, it can be chosen at random; there are other ways to reconfigure the hyperlinks after splitting or merging the pages; parameter values can be varied from one test to another; etc..

Simulations on a large number of random graph models with different implementations of the proposed algorithm have shown that the simple optimization scheme works quite well in continuously improving the cost J. To study the distribution of file transfers, let L^t be the random variable representing the transferred file sizes. Then for every data set (l_i, p_i), calculate the cumulative probability $P_i^t = \sum_{j \leq i} p_j$, with sizes of individual files ordered as $l_i \geq l_{i+1}$, i.e. $P_i^t = P[L^t \geq l_i]$.

The simulation results on a random Web site model with $N_{node} = 1000$ and $N_{link} = 2$ is shown in Figure 13.3, Figure 13.4 and Figure 13.5. Figure 13.3 illustrates the effectiveness of the naive splitting-merging heuristic in reducing the cost J iteration by iteration. The improvement is quite dramatic at the beginning, then it slows down and finally reaches some steady state after 350 iterations.

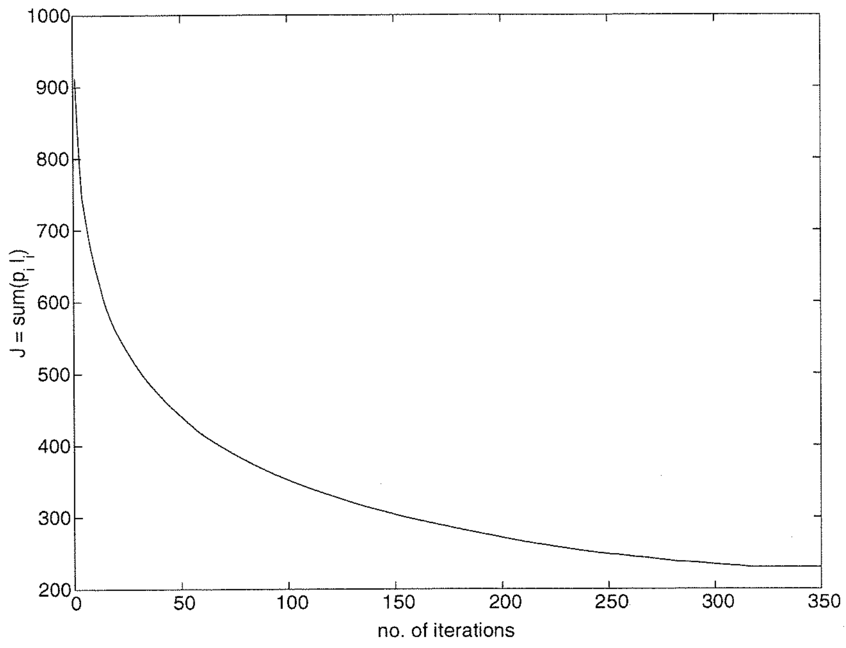

Figure 13.3: Cost function J vs. no of iterations, $N_{node} = 1000$, $N_{link} = 2$.

Figure 13.4 shows the cumulative distribution of file transfers P_i^t vs. l_i before and after the optimization. Motivated by the model (11.7) for the noncumulative p_i in Chapter 11, the following hyperbolic function

$$P_i^t = c_1(l_i + c_2)^{-1/\beta} \tag{13.5}$$

was used to fit the cumulative distribution P_i^t after the optimization. Note that the exponent in the equation becomes $-1/\beta$ instead of $-(1 + 1/\beta)$ in (11.7) due to the integration. The resulting model (dashed line in Figure 13.4) with $c_1 = 7.85 \times 10^5$, $c_2 = 318.28$ and $\beta = 0.42$ well approximates the simulation result over many orders of magnitude in the file size, which verifies that the distribution after the optimization displays a power law tail. Although the resulting value for β implies an exponent $\alpha = 1/\beta > 2$, which means the distribution is not really heavy-tailed as defined in Chapter 10, it does have an upper tail that is much heavier than the initial distribution before the optimization, which drops off exponentially at large l_i. In fact, the optimization process pushes the whole distribution curve down at small and medium file sizes to reduce the average download time. As a tradeoff, the optimized

Web site would produce more large file transfers than the original Web site.

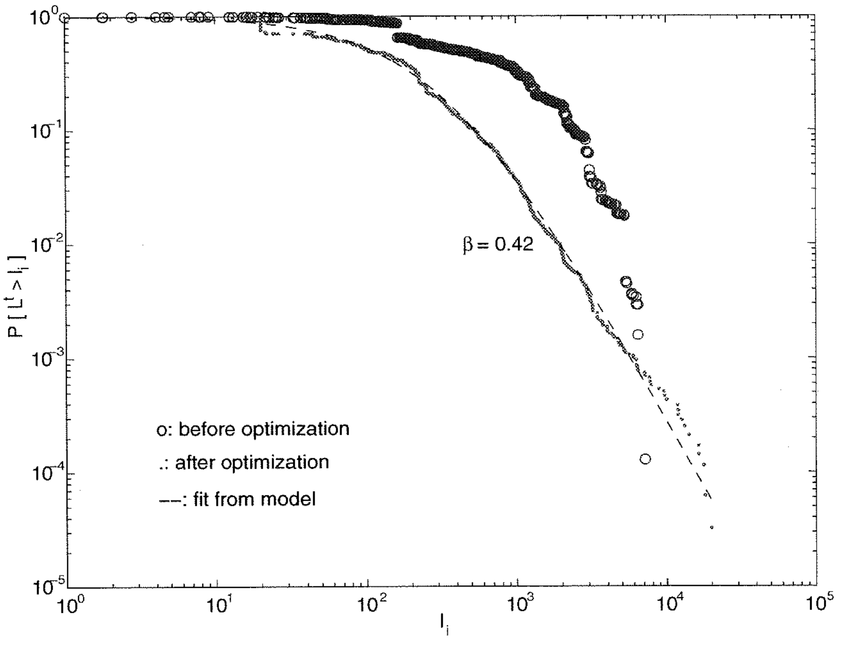

Figure 13.4: P_i^t vs. l_i for Web file transfers before optimization and after 350 iterations, $N_{node} = 1000$, $N_{link} = 2$.

Interestingly, we find that the distribution of the sizes of *unique* files on the Web site, as opposed to the traffic from the Web site, also has a nice power-law tail after the optimization, as displayed in Figure 13.5. Let L^u denote the random variable of unique file size on the Web site, then P_i^u denotes its complementary cumulative distribution, i.e., $P_i^u = P[L^u \geq l_i]$. Again the model in (13.5) can be used to fit P_i^u vs. l_i. The resulting parameters are $c_1 = 1.40 \times 10^6$, $c_2 = 1.04 \times 10^3$, and $\beta = 0.49$, indicating a "heavier" tail than the distribution for the transferred files. The heavy-tailed distribution for unique file sizes has also been observed from both the 1995 data and the 1998 data in [19, 9]. In [19] it was shown that the set of file transfers is intermediate in characteristics between the set of file requests and the set of unique files. This is due to the use of caching in Web applications. At one extreme, if no caching is provided, then the set of transferred files should be the same as the set of files that are requested, which is a simplification taken in this thesis. On the other hand, in the case of perfect (infinite) caching, the set of files transferred should be identical to the set of unique files. In reality, caching does exist but is never infinite,

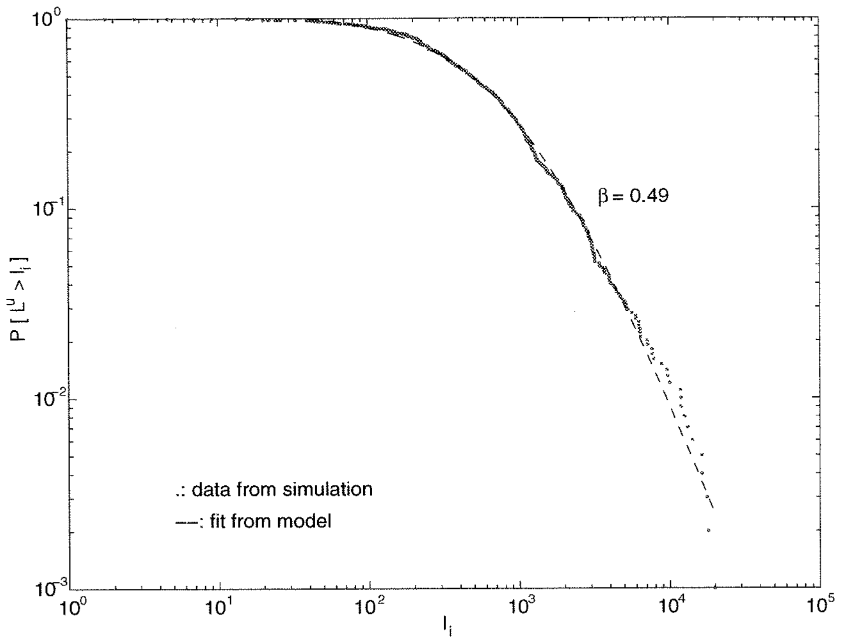

Figure 13.5: P_i^u vs. l_i for unique files after 350 iterations, $N_{node} = 1000$, $N_{link} = 2$.

so the distribution of resulting file transfers should be between the distributions of file requests and unique files. Back to the simulation results shown in Figure 13.4 and Figure 13.5, the distribution of L^t is less heavy-tailed than the distribution of L^u because small files get requested more often than large files. If caching is taken into account, then the distribution of file transfers should have a β value between 0.42 and 0.49 for this particular example. The effect of caching is not the focus of this thesis, but will be an interesting subject for future research.

These simulation results are quite robust with respect to the choices that can be made during the implementation of the optimization algorithm, as was discussed earlier in this subsection. Of course the objective function J may decay in different trajectories, and the resulting exponents of the power law tails can be slightly different, but the qualitative behavior of the same example of the Web layout model is fairly consistent in all the simulations.

Since the above example was generated at random, it may not be representative of the general characteristics of Web layouts of similar structure, say with the same number of N_{node} and N_{link}. Therefore, the above design and optimization procedure was also simulated on a subnet consisting of K random Web sites, each with $N_{node} = 1000$ and $N_{link} = 2$. All the Web sites are considered equally probable to be accessed. So the overall statistics are just the average of those for individual ones. After the simulation all K data sets of (l_i, p_i) were mixed together and the total cumulative distributions P_i^t of transferred files and P_i^u of unique files were computed. Both distributions contain power law tails, which are shown in the top two plots of Figure 13.6, compared with the distributions before the optimization that have exponential drop off at large file sizes. Again the hyperbolic model (13.5) was used to fit the simulation results, giving $\beta = 0.44$ for P_i^t and $\beta = 0.55$ for P_i^u. This means the distribution of transferred files has an exponent α slightly above 2 and the distribution of unique files has an exponent α slightly below 2. Comparing these exponents with the exponents obtained from the PLR model ($\alpha = 1$) and the one-dimensional Web layout model in Chapter 12 ($\alpha \approx 1$), it is easy to see that the distributions of file transfers from these graph models are less heavy-tailed than those from simpler models. This difference is consistent with the change in the empirically observed behavior of Web file transfers from the 1995 data to the 1998 data. A plausible explanation will be provided after more simulation results are presented.

The algorithm was also tested with different choices of N_{link}. Only the cumulative distributions of the transferred files L^t for $N_{link} = 1$(left) and $N_{link} = 3$(right) are displayed at the bottom of Figure 13.6. The distributions of the unique files L^u are not shown, but they have similar shape as in the $N_{link} = 2$ case (top right), and also are more heavy-tailed than the distributions of L^t with the same N_{link}. In the $N_{link} = 1$ case, P_i^t after the optimization can also be nicely fit using (13.5) with $\beta = 0.52$, indicating a power law tail that is heavier than that for $N_{link} = 2$ (top left). However, for P_i^t with $N_{link} = 3$, no perfect fit using (13.5) can be obtained since the distribution starts to drop off more quickly after $l_i \approx 10^5$. The best fit found leads to $\beta = 0.42$. This implies that the distribution for file transfers is even less heavy-tailed than for $N_{link} = 2$.

Comparison of these three cases suggests that as N_{link} becomes higher and higher, the tails of the cumulative distributions of transferred files become less and less heavy-

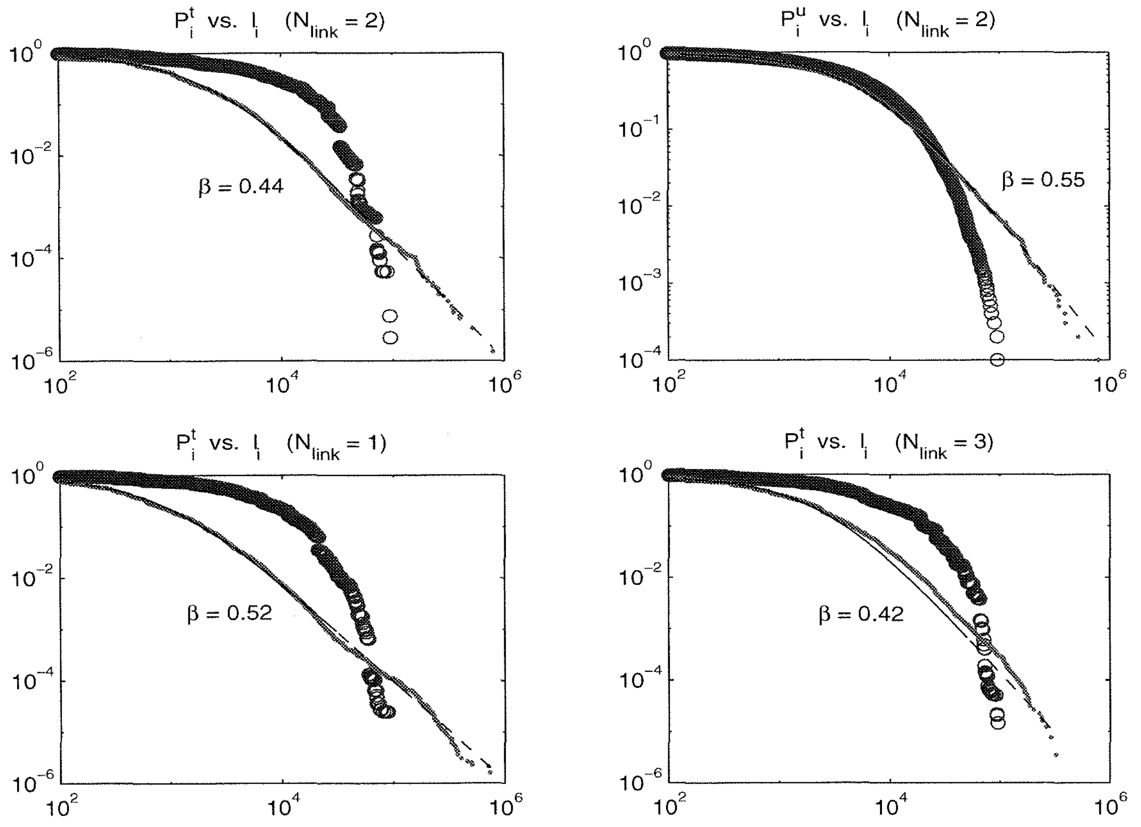

Figure 13.6: Mixed P_i^t vs. l_i for transferred files and mixed P_i^u vs. l_i for unique files from 10 random Web sites, each with $N_{node} = 1000$, before and after the optimization.

tailed. The parameter N_{link} captures the degree of connectivity of a Web site layout. The higher the N_{link}, the more connected the Web site. The consequence is the parameter β that fits the power law distribution of file transfers varies with the degree of connectivity. The PLR model characterizes one-dimensional Web layout with a very low degree of connectivity, $\beta = 1$; while the graph model for more connected Web layouts gives $\beta \approx 0.5$, and β becomes smaller as N_{link} increases. This observation provides a possible explanation for why the power law tail from the 1998 data was steeper than that from the 1995 data. The reason could be that from 1995 to 1998 the design of Web layout became more and more mature and sophisticated, including more extensive use of hyperlinks that led to greater connectivity in resulting Web sites, which, in turn, would demonstrate less heavy-tailed distributions in file transfers.

13.2.3 Two Special Cases

In the previous subsection, Web site topologies were modeled as incrementally generated graphs. This subsection studies two special cases of random graphs: chains and trees.

Chains

The simplest possible layout for a Web site is a chain-like structure, where all the Web pages are sequentially connected by unidirectional links, like the one in Figure 12.3. As in the general case, each Web page V_i contains a single file with length l_i, and the user enters the Web site at the first page V_1. At each page V_i, he has the choice of either leaving the Web site with probability p_e or following the hyperlink $E_{i,i+1}$ to browse the next page V_{i+1} with probability $1 - p_e$.

The splitting-merging iteration is quite trivial to implement in this special case. And the chain structure is preserved during the operations. Again small variations can be added to the standard algorithm, like in the case of general graphs. For example, split a file at a random location instead of in the middle, or pick p_e randomly in a certain range instead of at a fixed value. Simulations of different combinations of the above choices all give similar results. An example is demonstrated in Figure 13.7, where the cumulative distributions of the sizes of file transfers P_i^t vs. l_i before and after the optimization are shown. The distribution from the optimized Web site displays a very nice power law decay with an exponent $\alpha \approx 1$ in the whole range of l_i. Experiments of different N_{node} and p_e show that this slope is fairly independent of the parameter values, which validates the theoretical result from the one-dimensional Web layout model in Chapter 12.

Trees

Another special topology is the rooted-tree. It is a simplification of the implicit hierarchical structure that is present in many Web sites nowadays. The same incremental scheme for generating the initial topology in the random graph case is used for the

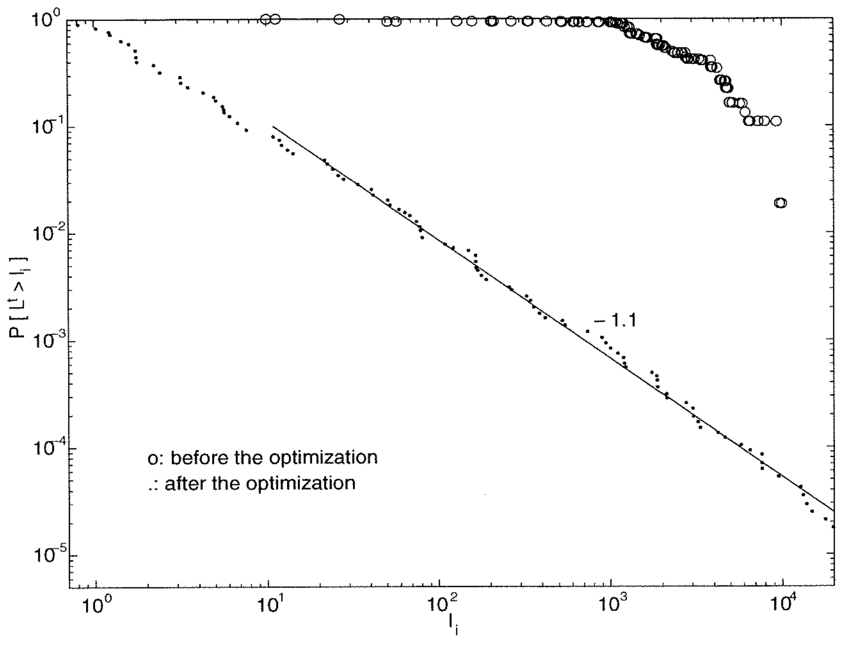

Figure 13.7: P_i^t vs. l_i for file transfers in a chain-like Web site before and after the optimization, $N_{node} = 100$.

tree structure, except that $N_{link} = 1$ and one more parameter needs to be specified: the maximum out-degree M_{out} for all the nodes. Note that M_{out} and N_{node} together determine the depth and the degree of connectivity of the initial tree.

The standard splitting-merging algorithm for the general graphs has to be modified so that the tree structure can be preserved during the optimization. The simulation result from one example is shown in Figure 13.8, which contains the cumulative distribution of sizes of file transfers P_i^t vs. l_i before and after the optimization. The distribution from the optimized Web site has a little curvature. It follows a power law with α close to 1 for small l_i, and the slope becomes steeper as the file size increases. A linear fit to the curve in the large l_i region gives $\alpha = 1.53$, which still suggests a heavy tail. Again this result in general does not depend on the specific values of the parameters N_{node} and M_{out}, although smaller M_{out} usually produces a heavier tail. A possible explanation for this is when N_{node} is fixed, a tree with smaller M_{out} spreads out less and goes down more deeply, which is closer to the one-dimensional chain case that produces a power law tail with $\alpha \approx 1$.

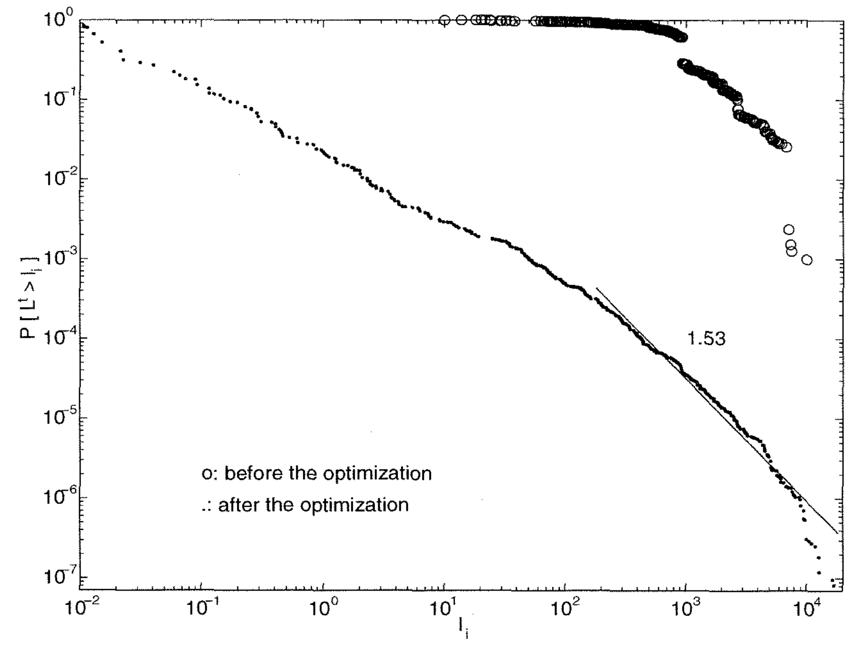

Figure 13.8: P_i^t vs. l_i of file transfers from a tree-like Web site before and after the optimization, $N_{node} = 500$, $M_{out} = 25$.

13.2.4 Summary

Comparison of the simulation results from chains, trees, and more general random graphs with different N_{link} suggests that the exponent of the power law tail in the distribution of file transfers from the optimized Web site is highly correlated with the initial degree of connectivity of the Web site. Among the three topologies that were studied, the chain structure has the least connectivity, the random graph with large N_{link} has the greatest connectivity, while the tree, similar to the random graph with $N_{link} = 1$,[2] is between those two extremes. At the same time, the exponents of the tails show a similar pattern. We could conclude that the more connected the graphs, the less heavy-tailed are the resulting distributions.

[2]The slight difference between these two is that in the tree case, the $N_{link} = 1$ property is preserved throughout the optimization.

13.3 A More Realistic Model

13.3.1 Generation of Web Page Geometry

The Web layout models presented in the previous section treat every Web page as an abstract node in a random graph without any geometry in it. Real Web pages are (mostly) HTML files containing embedded objects (images, etc.), distributed hyperlinks, and complicated layout. In this section a more realistic Web site model is introduced, which takes into account the internal structure of each Web page. Since it is based on the random graph model, only the new features that are not in the previous model are described here. So each page V_i still contains a single file with length l_i, but each file is divided into one or more paragraphs to coarsely capture the logical connection between information. Different paragraphs are considered relatively independent, and it is ok to split them and move them into different pages when necessary. The number of paragraphs in each file and the locations of the paragraph breaks are generated randomly. In addition, the hyperlinks that point out of this page can be located anywhere in the file. To avoid confusion, the model used in the previous section is referred to as the graph model, while the new model used in this section will be referred to as the geometric model.

Similar to the graph model, user navigation of the Web site always starts at page V_1. Inside each page V_i, assume that the user reads the file from top down. Again user interest $p(x)$ is assumed to be an exponential function of the location of the information x, with parameter λ, as has been studied in Chapter 12. λ can be easily adjusted to change the decay of user interest. When $\frac{1}{\lambda} \gg l_i$, then user interest is distributed almost uniformly throughout the whole page. If there is a hyperlink E_{ij} at location x_0, the user either follows the link to visit page V_j with probability P_n or keeps going in V_i. P_n is defined as the navigation probability, and treated as a constant throughout the whole Web site. Suppose there are k hyperlinks $E_{ij_1}, \ldots, E_{ij_k}$ in page V_i, and the probabilities of following these links are $p_{ij_1}, \ldots, p_{ij_k}$ respectively. Then $p_{i1} = 1 - \sum_{s=1}^{k} p_{ij_s}$ is defined to be the exiting probability.

The optimization algorithm for the geometric model is more involved than for the graph model, because we need to keep track of the paragraph structure and the locations of the hyperlinks in the original Web pages. Also locations of the cuts while splitting files are determined by assumptions on user interests, more specifically, on the parameter λ and how $\frac{1}{\lambda}$ is compared to l_i. Moreover, unless there is only one paragraph, the splitting point has to be between paragraphs. The relative locations of the hyperlinks in each paragraph will be preserved whenever files are split or merged. Every iteration of the splitting-merging procedure produces a new Markov chain, and there are no simple ways to obtain the new stationary probabilities p_i just by updating the old ones. So p_i will be recomputed using the updated transition probability matrix M at each iteration, which makes the computational cost higher than that of the graph model.

13.3.2 Simulation Results

A large number of simulations of the geometric model with different parameter settings have proven the effectiveness of the simple splitting-merging heuristic in reducing the cost function J, i.e., the average file download time from the Web site. One example of a Web site with 1000 nodes and $N_{link} = 2$ is demonstrated in Figure 13.9.

Figure 13.10 and Figure 13.11 show the cumulative distributions of sizes of transferred files (L^t) and unique files (L^u) from the same simulation as in Figure 13.9. Both distributions display power law tails, with $\beta = 0.52$ for P_i^t and $\beta = 0.81$ for P_i^u by fitting the hyperbolic model (13.5) to the simulation data. The fit here is not as good as those for the graph model because the distributions from the geometric model are less smooth than the previous ones. In addition, the β values from the fit are lower than those from the graph model, indicating that the distributions are more heavy-tailed. However, the qualitative behavior of the two models are fairly similar. In particular, the optimization process always reshapes the distribution of file transfers and results in a more heavy-tailed distribution than the original one without the optimization. Moreover, the distribution for the unique files is always

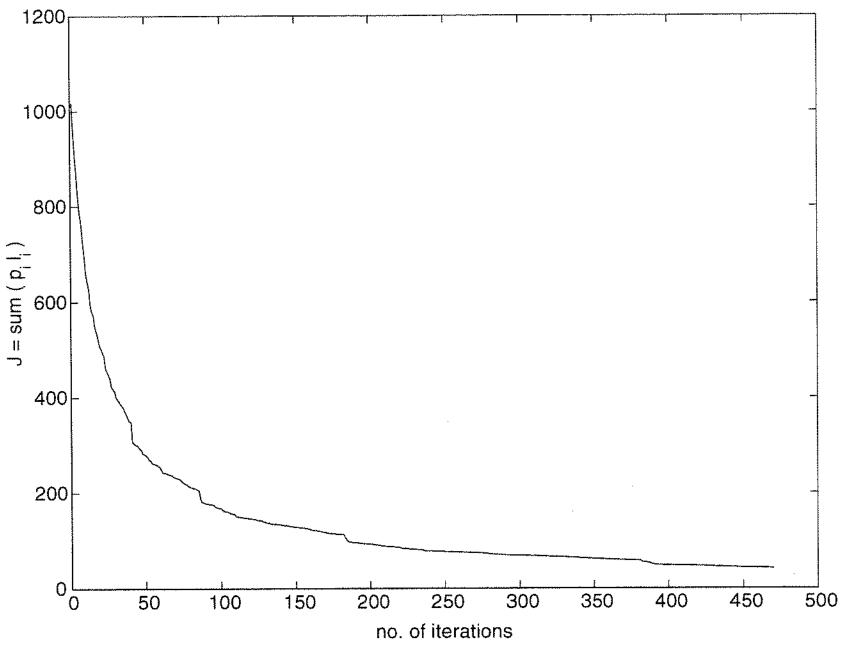

Figure 13.9: Cost function J vs. no. of iterations, $N_{node} = 1000$, $N_{link} = 2$.

more heavy-tailed than the distribution for the transferred files. And as was discussed in the previous section, if caching is taken into account, then the real distribution of the transferred files should be something between the P_i^t and P_i^u obtained from this model.

The simulation was also conducted on a subnet consisting of K random Web sites with page geometries, where each Web site has $N_{node} = 500$ and $N_{link} = 2$. The probabilities of visiting individual Web sites are assumed to be equal. The results are shown on the top two plots of Figure 13.12. Again both cumulative distributions for the mixed L^t and L^u contain power law tails and the tail of P_i^u ($\beta = 0.89$) is heavier than that of P_i^t ($\beta = 0.58$). The cumulative distributions of the transferred files from two other simulations with $N_{link} = 1$ and $N_{link} = 3$ are shown in the bottom two plots of Figure 13.12. Both have power law tails with $\beta = 0.58$ and $\beta = 0.56$ respectively. Although the fit using (13.5) is not as good as those for the graph model, as with the single Web site case, it still captures the basic shape of the distributions. However, with the geometric model, the value of β from the fit is fairly stable for Web layouts with different N_{link}, which is not the case with the graph model. It seems that the

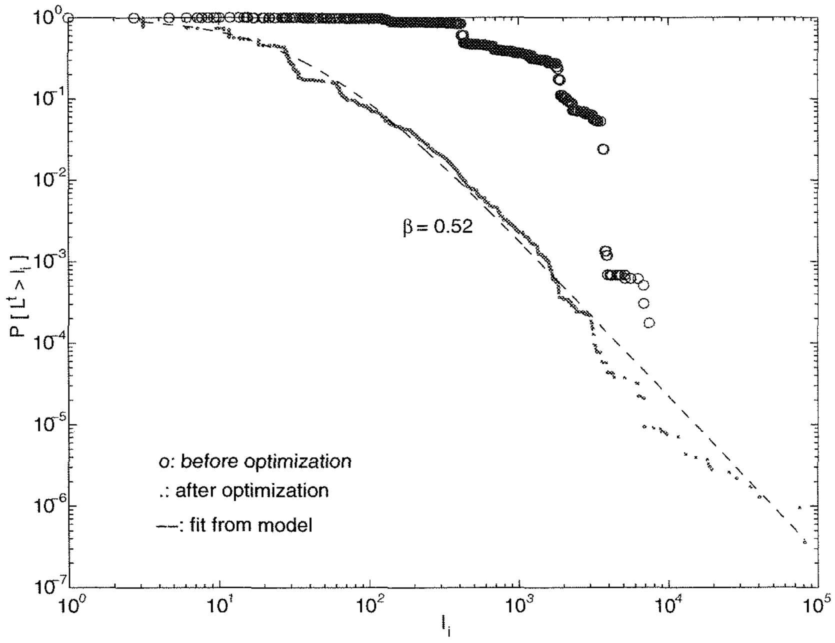

Figure 13.10: P_i^t vs. l_i for Web file transfers before optimization and after 470 iterations, $N_{node} = 1000$, $N_{link} = 2$.

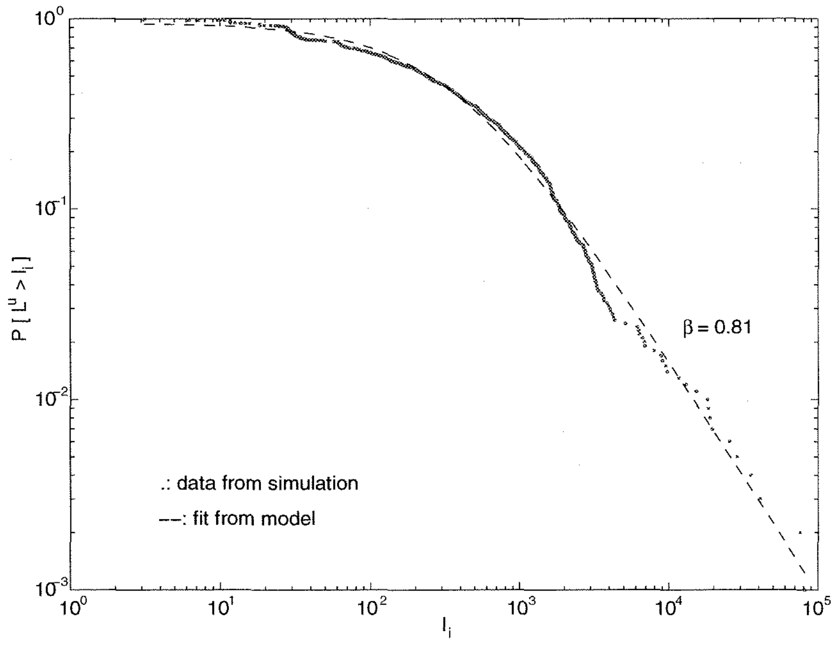

Figure 13.11: P_i^u vs. l_i for Web file transfers after 470 iterations, $N_{node} = 1000$, $N_{link} = 2$.

internal structure of individual Web pages blurred the difference in the connectivity caused by the difference in the number of links. A full understanding of its causes and implications will require more simulations with different kinds of Web layout models, and more importantly, a combination of simulations and an empirical study of real Web sites. What is consistent between the graph model and the geometric model is that the resulting distributions for Web file transfers all have power law tails with exponent $\alpha \approx 2$, or $\beta \approx 0.5$ in the hyperbolic model (13.5). This is much less heavy-tailed than the distributions from the PLR model or the one-dimensional Web layout model where $\beta \approx 1$. The relation between this simulation result and empirical data from [19, 9] as well as a possible explanation have been given in the previous section with the graph model, hence will not be repeated here.

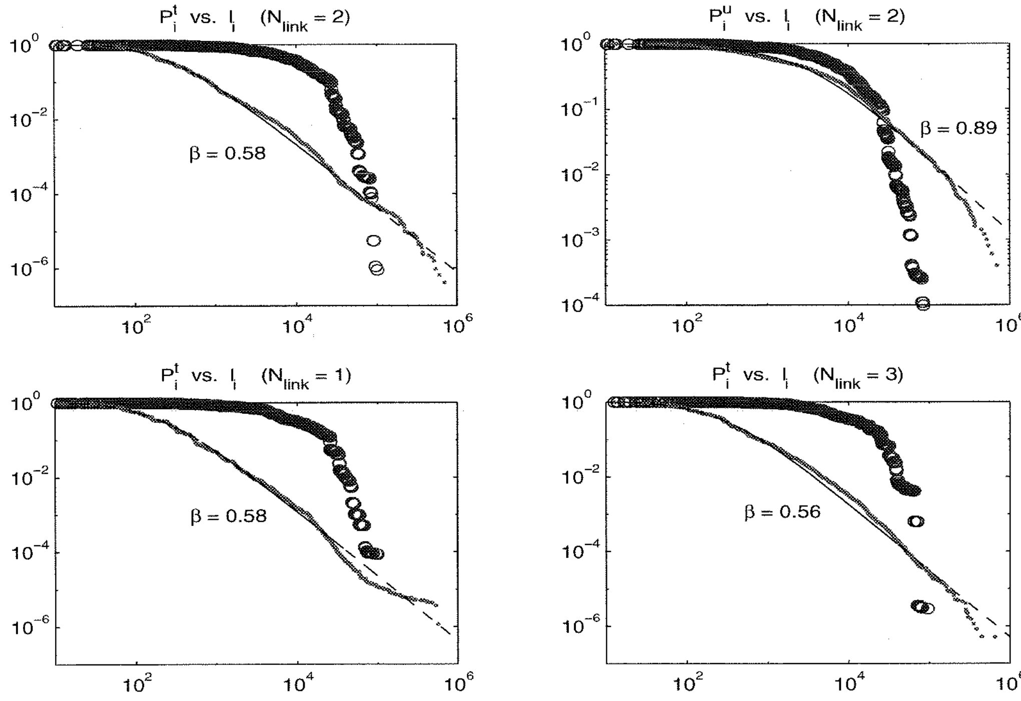

Figure 13.12: Mixed P_i^t vs. l_i for transferred files and mixed P_i^u vs. l_i for unique files from 10 random Web sites, each with $N_{node} = 500$, before and after the optimization.

13.4 Future Improvements of Graph-Based Models

Although the geometric model has a higher level of realism than the simple graph model, it is still an abstraction of real Web sites nowadays. Although we do believe that these models capture some important features of Web layout design that provide a plausible explanation of the origin of heavy-tailed distributions of Web file transfers, there are many more structures and characteristics of real Web sites that are not yet included in our abstract models. The following are some of the issues that are worth looking into.

First of all, the Markov chain model for user navigation, which assumes that the probability a user takes a certain hyperlink E_{ij} from page V_i is independent of the pages the user has visited before V_i, is a simplification of perhaps more intricate user behavior. The validity of this approximation needs to be verified by empirical studies of user access patterns on real Web sites. In the case when user navigation demonstrates strong memory so that the Markov chain assumption cannot be used, the idea proposed in [34] that the average probability that a user visits the next page decays with the number of pages this user has already visited during each Web session can be adopted to modify our model. With this assumption a fixed transition probability matrix M does not exist, so the probabilities p_i of each Web page being visited cannot be calculated directly. However, by repeatedly simulating user access to the Web site, p_i can be well approximated by the average hit rate $\hat{p}_i$ of each page as long as the number of simulations run is sufficiently large.

Second, both of the graph models we have discussed did not distinguish between text (HTML) files and multimedia such as images, audio and video files. When these files are embedded in the HTML files, they are treated as part of a large file so that they can be reorganized during the splitting and merging procedure. However, multimedia files are hard to split and merge. The paragraph structure in the geometric model captures this feature to certain degree. But since each embedded object has a unique URL and contributes to the Web traffic as an individual file, it is desirable to incorporate them into our graph model so that their impact on the heavy-tailed distributions of Web file transfers can be evaluated. This is especially important since the traffic generated by real-time multimedia applications has become an increasingly larger contributor to the overall Web traffic. One thing worth pointing out is that according to a study by Crovella et al. in [19] on the relationship between distributions of different types of files, although the presence of multimedia formats does add to the weight of the tail in the overall distribution, the distribution of text files itself is heavy-tailed.

Finally, so far we have taken a relatively static view of files on the Web. With the fast development in Web technologies, Web contents nowadays become more and more dynamic. Many Web files are generated on real time when a user is browsing the Web site. It will be a challenging and interesting task for future research to study the impact of dynamic contents on Web site layout design and heavy-tailed Web traffic.

Chapter 14 Concluding Remarks

14.1 Summary and Discussion

This thesis has presented a series of models of Web design which treated the layout of a Web site as an optimal design problem. The design objective was assumed to be the minimization of the average latency that the user experiences in downloading files while browsing the Web site, subject to constraints on the total number of files. For simplicity and to make contact with the data collected by Crovella et al. in 1995, when the WWW just started to take off, it was assumed that Web sites in their earlier stage consisted of preexisting one-dimensional documents that were simply split into files and linked trivially. With either the PLR model in Chapter 11 or the one-dimensional Web layout model in Chapter 12, it was found consistently that the resulting distribution of Web file transfers had a power law tail with exponent $\alpha \approx 1$. In Chapter 13 more sophisticated Web layouts were explored, which viewed general Web sites as random graphs and simulated the design process using heuristic optimization schemes. Again the resulting distribution of transferred files from the optimized Web site displays a power law tail with exponent $\alpha \approx 2$. The observation that the distributions of Web file transfers become less heavy-tailed with more complex Web layouts is also consistent with the change from the 1995 data to the new data collected in 1998. An implication of this result is that optimal Web layouts and more effective use of hyperlinks may tend to produce much less bursty network traffic, which will be an interesting subject in our future study.

Does the idealized problem described here actually explain the data, or is the remarkable agreement simply a coincidence? Even more important questions are what are the implications of these results for Web design, network traffic, network performance, and protocol design? We will attempt to give tentative answers, but since this work is relatively new, we must caution that any answers are necessarily fairly speculative.

It is unlikely that early (or current) Web designers had any explicit intent to optimally design the layout of their Web sites, so we would expect individual Web sites to deviate from optimal to varying degrees. It is likely however that even the most ad hoc approach to Web layout would naturally make small home pages, with links to increasingly larger files as one moved deeper into the Web site. Furthermore, the natural layout of documents that predates the Web involves hierarchical structures with heavy tails, with titles, abstracts, introductions, summaries, reviews, chapters, appendices, and so on. The 1995 data was an aggregation of a large number of file downloads from a variety of Web sites, none of which are necessarily optimal, but most of which are probably reasonably well designed. Thus it is not surprising that the aggregate behavior might have statistics that appear optimal even if the individual Web sites are not. Furthermore, users would tend to avoid very poorly designed Web sites, adding an element of selection to the data.

Most importantly, even if the agreement with data is entirely coincidental, it is still true that Web sites *should* have these statistics if they are designed to minimize average file download times. This has more serious implications than the agreement with data. In particular, heavy tailed distributions can be viewed as the inevitable outcome of a very natural optimal "source coding" problem that is analogous to standard data compression, but with very different resulting distributions. Given the connection of heavy tails in Web sites and bursty network traffic, this can be thought of as bringing some initial closure to the origins of such traffic, but raises new questions. Bursty traffic is thus not an artifice of user behavior, but has some aspects which are intrinsic to at least the current dominant application, the WWW, and may be even more intrinsic to any application which organizes information for human consumption.

14.2 Future Directions

As discussed at the end of Chapter 13, the Web layout models studied in this thesis are still at a high level of abstraction from real Web sites today. More realistic models of Web layouts with increasing levels of complexity can be built on top of these relatively simplified models. It is worth the effort to gather empirical data of real Web site topology and user access patterns so that the truthfulness of the simulation models can be tested and enhanced. Moreover, there are many more design objectives and resource constraints other than minimizing the average download time and limiting the total number of files on a Web site. Alternative settings for the optimization problem need to be explored to see whether similar conclusions can be drawn.

Additionally, if Web site layout can be viewed as "source coding" albeit with many strange and unfamiliar properties, then network protocol design and congestion control might profitably be viewed as a form of channel coding, but also presumably with strange and unfamiliar properties. There may be some advantages in exploiting the specific features of the resulting source, as well as advantages in some joint design where the source coding reflects the nature of the network on which it must be transmitted. Recent work along these lines include designing new scheduling policies on Web servers for specifically dealing with heavy-tailed workloads [33, 20]. It may also be possible to utilize the knowledge of the heavy-tailed Web traffic by making more effective use of caching and prefetching at the Web application level. A good example of combining source coding with channel coding is the rate distortion theory in data compression [27]. There is great potential in combining rate distortion techniques and Web layout design for more efficient transmission of information over the Internet. Another promising direction is to integrate generalized source coding with Low et al.'s optimization flow control theory [44, 5] so that joint source and channel coding can be viewed as a global optimization problem. Hopefully this mixed framework can facilitate a new look into different TCP congestion control algorithms under self-similar network traffic in order for possible updates of existing protocols or design of new protocols to be explored.

Appendix A Proof of the Asymptotic Results in Chapter 12

Consider the ideal case where $L = \infty$. Let $F(x)$ be the CCDF corresponding to the PDF $p(x)$, i.e., $F(x) = \int_x^\infty p(x)dx$. The probability of transferring file #i, whose size is $l_i = c_i - c_{i-1}$, is $p_i^t = F(c_{i-1})$. Based on the recursion relation $p(c_i)l_{i+1} = p_i$, where $p_i = \int_{c_{i-1}}^{c_i} p(x)dx$, we want to show that

$$p_i^t \sim k l_i^{-1} \tag{A.1}$$

as $l_i \to \infty$ for certain $p(x)$, where $k > 0$ is a constant independent of l_i. This is equivalent to

$$\frac{\log p_{i+1}^t - \log p_i^t}{\log l_{i+1} - \log l_i} \sim -1. \tag{A.2}$$

In general, we have

$$\frac{\log p_{i+1}^t - \log p_i^t}{\log l_{i+1} - \log l_i} = \frac{\log F(c_i) - \log F(c_{i-1})}{\log \frac{p_i}{p(c_i)} - \log l_i} = \frac{\log F(c_i) - \log F(c_{i-1})}{\log \frac{F(c_{i-1}) - F(c_i)}{p(c_i) l_i}}.$$

Therefore, for any $p(x)$ that satisfies

$$\frac{\log F(c_i) - \log F(c_{i-1})}{\log \frac{F(c_{i-1}) - F(c_i)}{p(c_i) l_i}} \sim -1, \tag{A.3}$$

as $l_i \to \infty$ (c_i, $c_{i-1} \to \infty$), (A.1) holds. Although it is hard to find the exact class of distributions that satisfy (A.3), we can use this condition to test individual distributions. The case where $p(x)$ is an exponential distribution was shown in Chapter 12. Next we prove that (A.3) holds for the normal distribution and the Cauchy distribution as well.

A.1 For the Gaussian Distribution

We assume that $p(x)$ is one side of a normal distribution, and we only show for the standardized case. For more general cases, a similar argument holds. Here $p(x) = \frac{2}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$, $x \geq 0$.

As $x \to \infty$, we have $F(x) \sim \frac{2}{\sqrt{2\pi x}}e^{-\frac{x^2}{2}}$. Then as $l_i \to \infty$,

$$\log F(c_i) - \log F(c_{i-1}) \sim \log \frac{e^{-\frac{c_i^2}{2}}}{\sqrt{2\pi c_i}} - \log \frac{e^{-\frac{c_{i-1}^2}{2}}}{\sqrt{2\pi c_{i-1}}} = -\frac{1}{2}(c_i^2 - c_{i-1}^2) + \log c_{i-1} - \log c_i,$$

and

$$\log \frac{F(c_{i-1}) - F(c_i)}{p(c_i)l_i} \sim \log \frac{\frac{1}{\sqrt{2\pi c_{i-1}}}e^{-\frac{c_{i-1}^2}{2}} - \frac{1}{\sqrt{2\pi c_i}}e^{-\frac{c_i^2}{2}}}{\frac{1}{\sqrt{2\pi}}e^{-\frac{c_i^2}{2}}l_i} = \log[\frac{1}{c_i}(\frac{c_i}{c_{i-1}}e^{\frac{c_i^2-c_{i-1}^2}{2}} - 1)] - \log l_i$$

$$\sim \log \frac{1}{c_{i-1}}e^{\frac{c_i^2-c_{i-1}^2}{2}} - \log l_i = \frac{1}{2}(c_i^2 - c_{i-1}^2) - \log c_{i-1} - \log l_i.$$

Hence,

$$\frac{\log F(c_i) - \log F(c_{i-1})}{\log \frac{F(c_{i-1})-F(c_i)}{p(c_i)l_i}} \sim \frac{-\frac{1}{2}(c_i^2 - c_{i-1}^2) + \log c_{i-1} - \log c_i}{\frac{1}{2}(c_i^2 - c_{i-1}^2) - \log c_{i-1} - \log l_i} \sim -1.$$

A.2 For the Cauchy Distribution

Again for simplicity assume $\lambda = 1$. So $p(x) = \frac{2}{\pi(1+x^2)}$, $x \geq 0$.

As $x \to \infty$, we have $p(x) \sim \frac{2}{\pi}x^{-2}$, and $F(x) = 1 - \frac{2}{\pi}\arctan x \sim \frac{2}{\pi}x^{-1}$. Then as $c_i,\ c_{i-1} \to \infty$,

$$l_{i+1} = \frac{p_i}{p(c_i)} = \frac{F(c_{i-1}) - F(c_i)}{p(c_i)} \sim \frac{\frac{2}{\pi}c_{i-1}^{-1} - \frac{2}{\pi}c_i^{-1}}{\frac{2}{\pi}c_i^{-2}} = c_i(\frac{c_i}{c_{i-1}} - 1) = \frac{c_i}{c_{i-1}}l_i.$$

Hence,

$$\frac{\log F(c_i) - \log F(c_{i-1})}{\log \frac{F(c_{i-1})-F(c_i)}{p(c_i)l_i}} = \frac{\log F(c_i) - \log F(c_{i-1})}{\log c_i - \log c_{i-1}} \sim -1.$$

A by-product of this proof is the following:

$$\frac{l_{i+1}}{l_i} \sim \frac{c_i}{c_i - 1} \Longrightarrow \frac{l_{i+1}}{l_i} \sim \frac{c_i + l_{i+1}}{c_{i-1} + l_i} = \frac{c_{i+1}}{c_i}.$$

Additionally

$$\frac{l_{i+2}}{l_{i+1}} \sim \frac{c_{i+1}}{c_i},$$

therefore,

$$\frac{l_{i+1}}{l_i} \sim \frac{l_{i+2}}{l_{i+1}} \sim \cdots = \text{constant}.$$

The above argument explains why the optimal l_i in the Cauchy case are distributed uniformly on a logarithmic scale, as shown in Figure 12.5.

A.3 Properties of the Operator '$\sim$'

The operator '$\sim$' defines an equivalent class of functions. In particular, if $f(x) \sim g(x)$ as $x \to a$, then $\lim_{x\to a} \frac{f(x)}{g(x)} = 1$, where $a = 0$, or ∞. [65]

The following are some useful properties of the equivalent class defined by '$\sim$'.

1. $f(x) \sim g(x) \Longleftrightarrow \frac{f(x)}{g(x)} \sim 1.$

2. $f(x) \sim g(x) \Longleftrightarrow g(x) \sim f(x).$

3. $f(x) \sim g(x)$ and $g(x) \sim h(x) \Longrightarrow f(x) \sim h(x).$

4. $f(x) \sim g(x) \Longrightarrow h(x)f(x) \sim h(x)g(x)$, for $h(x) \neq 0$.

5. $f_1(x) \sim g_1(x)$ and $f_2(x) \sim g_2(x) \Longrightarrow \frac{f_1(x)}{f_2(x)} \sim \frac{g_1(x)}{g_2(x)}.$

6. If $f_1(x) \sim g_1(x)$ and $f_2(x) \sim g_2(x), g_1(x) + g_2(x) \neq 0$, and either $\lim_{x\to a} \frac{g_1(x)}{g_2(x)}$ or $\lim_{x\to a} \frac{g_2(x)}{g_1(x)}$ exists, then $f_1(x) + f_2(x) \sim g_1(x) + g_2(x)$.

7. If $f(x) \sim g(x)$, and $\exists c > 0$, such that $|\log g(x)| \geq c$, then $\log f(x) \sim \log g(x)$.

Bibliography

[1] R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the World-Wide Web. In *Nature*, volume **401(6749)**, pages 130–131, 1999.

[2] G. Apostolakis. The concept of probability in safety assessments of technological systems. In *Science*, volume **250(4986)**, pages 1359–1364, 1990.

[3] M.F. Arlitt and C.L. Williamson. Web server workload characterization: The search for incariants. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 126–137, 1996.

[4] B. Arnold. *Pareto Distributions.* International Cooperative Publishing House, Maryland, 1983.

[5] S. Athuraliya, D.E. Lapsley, and S.H. Low. An enhanced random early marking algorithm for Internet flow control. In *Proceedings of Infocom '2000, Isreal*, 2000. To appear.

[6] P. Bak. *How Nature Works: The Science of Self-Organized Criticality.* Copernicus, New York, 1996.

[7] V. Balakrishnean, S. Boyd, and S. Balemi. Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems. In *International Journal of Robust and Nonlinear Control*, volume **1**, pages 295–317, 1991.

[8] A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: The topology of the world wide web. In *Preprint submitted to Elsevier Preprint*, 1999.

[9] P. Barford, A. Bestavros, A. Bradley, and M.E. Crovella. Changes in web client access patterns: Characteristics and caching implications,. In *World Wide Web:*

Special Issue on Characterization and Performance Evaluation, volume **12**, pages 15–18, 1999.

[10] B. R. Barmish and C. M. Lagoa. The uniform distribution: A rigorous justification for its use in robustness analysis. In *Mathematics of Control, Signals and Systems*, volume **10**, pages 203–222, 1997.

[11] C. Beck and J.C. Doyle. Mixed μ upper bound computation. In *Proceedings of the IEEE Conference on Decision and Control*, volume **4**, pages 3187–3192, 1992.

[12] R.P. Braatz, P.M. Young, J.C. Doyle, and M. Morari. Computational complexity of μ calculation. In *IEEE Transactions on Automatic Control*, volume **39**, pages 1000–1002, 1994.

[13] F. Brichet, J. Roberts, A. Simonian, and D. Veitch. Heavy traffic analysis of a storage model with long range dependent on/off sources. In *Queueing Systems*, volume **23(1-4)**, pages 197–215, 1996.

[14] J.M. Carlson and J.C. Doyle. Highly Optimized Tolerance: A mechanism for power laws in designed systems. In *Physics Review E*, volume **60**, pages 1412–1428, 1999.

[15] J.M. Carlson and J.C. Doyle. Highly Optimized Tolerance: Robustness and design in complex systems. In *Physics Review Letters*, volume **84(11)**, pages 2529–2532, 2000.

[16] J. Chen, M.K.H. Fan, and C.N. Nett. Structured singular values and stability analysis of uncertain polynomials (Part I): The generalized μ. In *Systems and Control Letters*, volume **23**, pages 53–65, 1994.

[17] H. Chernoff. A measure of asymptotic efficiency for test of hypothesis based on the sum of observations. In *Annals of Mathematical Statistics*, volume **23**, pages 493–507, 1952.

[18] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* John Wiley and Sons, New York, 1991.

[19] M.E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. In *IEEE/ACM Transactions on Networking*, volume **5(6)**, pages 835–846, 1997.

[20] M.E. Crovella, R. Frangioso, and M. Harchol-Balter. Connection scheduling in Web servers. In *Proceedings of the 1999 USENIX Symposium on Internet Technologies and Systems, Boulder, Coloraro*, 1999.

[21] P. Danzig, S. Jamin, R. Cáceres, D. Mitzel, and D. Estrin. An empirical workload model for driving wide-area TCP/IP network simulations. In *Internetworking: Research and Experience*, volume **3**, pages 1–26, 1992.

[22] R.R.E. de Gaston and M.G. Safanov. Exact calculation of the multiloop stability margin. In *IEEE Transactions on Automatic Control*, volume **33**, pages 156–171, 1988.

[23] M.B. Doar. A better model for generating test networks. In *Proceedings of Globecom '96*, 1996.

[24] J.C. Doyle. Analysis of feedback systems with structured uncertainty. In *IEE Proceedings*, volume Part D, **129(6)**, pages 242–250, 1982.

[25] J.C. Doyle and J.M. Carlson. Highly Optimized Tolerance and generalized source coding. In *Physics Review Letters*, 2000. submitted.

[26] N.G. Duffield and N. O'Connell. Large deviations and overflow probabilities for the general single-server queue, with applications. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume **118**, pages 363–374, 1995.

[27] M. Effros. Distortion-rate bounds for fixed- and variable-rate multiresolution source codes. In *IEEE Transactions on Information Theory*, volume **45(6)**, pages 1887–1910, 1999.

[28] A. Erramilli, O. Narayan, and W. Willinger. Experimental queueing analysis with long-range dependent packet traffic. In *IEEE/ACM Transactions on Networking*, volume **4(2)**, pages 209–223, 1996.

[29] B.S. Everitt and D.J. Hand. *Finite Mixture Distributions.* Chapman and Hall, London, New York, 1981.

[30] M.K.H. Fan, A.L. Tits, and J.C. Doyle. Robustness in the presence of mixed parametric uncertainty and unmodeled dynamics. In *IEEE Transactions on Automatic Control*, volume **36**, pages 25–38, 1991.

[31] M Fu. The real structured singular value is hardly approximable. In *IEEE Transactions on Automatic Control*, volume **42(9)**, pages 1286–1288, 1997.

[32] M. Garrett and W. Willinger. Analysis, modeling, and generation of self-similar VBR video traffic. In *Proceedings of SIGCOMM '94*, pages 269–280, 1994.

[33] M. Harcol-Balter, M. Crovella, and S.-S. Park. The case for SRPT scheduling in Web servers. In *MIT-LCS-TR-767*, 1998.

[34] B.A. Huberman, P.L.T. Pirolli, J.E. Pitkow, and R.M. Lukose. Strong regularities in World Wide Web surfing. In *Science*, volume **280(5360)**, pages 95–97, 1998.

[35] G. Irlam. ufs'93 [Updated file size survey results]. In *USENET newsgroup comp.os.research, message 2ddp3b$jn5@darkstar.ucsc.edu*, Nov. 29, 1993.

[36] R. Jain and S.A. Routhier. Packet trains: Measurements and a new model for computer network traffic. In *IEEE Journal on Selected Areas in Communications*, volume **4**, pages 986–995, 1986.

[37] Park K., G.T. Kim, and M.E. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proceedings of 4th International Conference in Network Protocols*, pages 171–180, 1996.

[38] S. Khatri and P. Parrilo. Guaranteed bounds for probabilistic μ. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3349–3354, 1998.

[39] S. Khatri and P. Parrilo. Spherical μ. In *Proceedings of the American Control Conference*, pages 2314–2318, 1998.

[40] W.E. Leland and T.J. Ott. Unix process behavior and load balancing among loosely-coupled computers. In O.J. Boxman, J.-W. Cohen, and H.C. Tijms, editors, *Teletraffic Analysis and Computer Performance Evaluation*, pages 191–208, Amsterdam, 1986. Elsevier Science Publishers B. V.

[41] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. On the self-similar nature of Ethernet traffic (extended version). In *IEEE/ACM Transactions on Networking*, volume **2(1)**, pages 1–15, 1994.

[42] W.E. Leland and D.V. Wilson. High time-resolution measurement and analysis of LAN traffic: Implications for LAN interconnection. In *Proceedings of IEEE INFOCOM, Bal Harbour, FL*, pages 1360–1366, 1991.

[43] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering.* Addison-Wesley, 1994.

[44] S.H. Low and D.E. Lapsley. Optimization flow control I: Basic algorithm and convergence. In *IEEE/ACM Transactions on Networking*, volume **7(6)**, pages 861–874, 1999.

[45] B.B. Mandelbrot. Long-run linearity, locally gaussian processed, h-spectra and infinite variances. In *International Economic Review*, volume **10**, pages 82–113, 1969.

[46] B.B. Mandelbrot. *The Fractal Geometry of Nature.* Freeman, New York, 1983.

[47] B.B. Mandelbrot and J.R. Wallis. Noah, Joseph, and operational hydrology. In *Water Resources Research*, volume **4**, pages 909–918, 1968.

[48] K. Meier-Hellstern, P.E. Wirth, Y.-L. Yan, and D.A. Hoeflin. Traffic models for ISDN data users: Office automation application. In A. Jensen and V.B. Iversen, editors, *Teletraffic and Datatraffic in a Period of Change, Proc. of ITC13, Copenhagen*, pages 167–172, Amsterdam, 1991. Elsevier Science Publishers B. V.

[49] A.O. Mendelzon and T. Milo. Formal models of Web queries. In *Proceedings of the Sixteenth ACM Symposium on Principles of Database Systems, Tucson, Arizona*, 1997.

[50] Moore N., D. Ebbeler, and M. Creager. A methodology for probabilistic prediction of structural failures of launch vehicle propulsion systems. In *AIAA/ASME/ASCE/AHS/ASC 31st Annual Structures, Structural Dynamics and Materials Conference*, Long Beach, 1990.

[51] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming.* SIAM publications, 1994.

[52] M.P. Newlin. *Model Validation, Control, and Computation.* PhD thesis, California Institute of Technology, 1996.

[53] M.P. Newlin and S. Glavaski. Advances in the computation of the μ lower bound. In *Proceedings of the American Control Conference*, pages 442–446, 1995.

[54] M.P. Newlin and P.M. Young. Mixed μ problems and branch and bound techniques. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3175–3180, 1992.

[55] I. Norros. A storage model with self-similar input. In *Queueing Systems*, volume **16**, pages 387–396, 1994.

[56] A.K. Packard and J.C. Doyle. The complex structured singular value. In *Automatica*, volume **29**, pages 71–109, 1993.

[57] F. Paganini. *Sets and Constraints in the Analysis of Uncertain Systems.* PhD thesis, California Institute of Technology, 1996.

[58] P. Parrilo and S. Khatri. Closed form solutions for a class of lmis. In *Proceedings of the American Control Conference*, pages 87–91, 1998.

[59] V. Paxson. Empirically-Derived analytic models of wide-area TCP connections. In *IEEE/ACM Transactions on Networking*, volume **2**, pages 316–336, 1994.

[60] V. Paxson and S. Floyd. Wide-area traffic: the failure of poisson modeling. In *IEEE/ACM Transactions on Networking*, volume **3(3)**, pages 226–244, 1995.

[61] V. Paxson and S. Floyd. Why we don't know how to simulate the Internet. In *Proceedings of the 1997 Winter Simulation Conference*, 1997.

[62] J.E. Pitkow and C.M. Kehoe. GVU's WWW Users Surveys [online]. 1997. Available at www.gvu.gatech.edu/user_surveys.

[63] L.R. Ray and R.F. Stengel. A Monte Carlo approach to the analysis of control systems robustness. In *Automatica*, volume **29**, pages 229–236, 1993.

[64] M.G. Safonov. Stability margins of diagonally perturbed multivariable feedback systems. In *IEE Proceedings*, volume Part D, **129(6)**, pages 251–256, 1982.

[65] G. Samorodnitski and M.S. Taqqu. *Stable Non-Gaussian Random Process: Stochastic Models with Infinite Variance.* Chapman and Hall, New York, 1994.

[66] C.E. Shannon. A mathematical theory of communication. In *Bell System Technical Journal*, volume **27(3)**, pages 379–423, 623–656, 1948.

[67] A. Sideris and R.S. Sánchez Peña. Robustness margin calculations with dynamic and real parametric uncertainty. In *IEEE Transactions on Automatic Control*, volume **35**, pages 970–974, 1990.

[68] M.S. Taqqu, W. Willinger, and R. Sherman. Proof of a fundamental result in self-similar traffic modeling. In *Computer Communication Review*, volume **27**, pages 5–23, 1997.

[69] R. Tempo, E.W. Bai, and F. Dabbene. Probabilistic robustness analysis: Explicit bounds for the minimum number of samples. In *Systems and Control Letters*, volume **30**, pages 237–242, 1997.

[70] H.A. Wan and C.-W. Chang. Web page design and network analysis. In *Internet Research: Electronic Networking Applications and Policy*, volume **8(2)**, pages 115–122, 1998.

[71] B.M. Waxman. Routing of multipoint connections. In *IEEE Journal on Selected Areas in Communications*, pages 1617–1622, 1988.

[72] W. Willinger and V. Paxson. When mathematics meets the Internet. In *Notices of the AMS*, volume **45(8)**, pages 961–970, 1998.

[73] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson. Self-similarity through high variability: statistical analysis of Ethernet LAN traffic at the source level. In *IEEE/ACM Transactions on Networking*, volume **5(1)**, pages 71–86, 1997.

[74] P.M. Young. *Robustness with Parametric and Dynamic Uncertainty.* PhD thesis, California Institute of Technology, 1993.

[75] P.M. Young, M.P. Newlin, and J.C. Doyle. Practical computation of the mixed μ problem. In *Proceedings of the American Control Conference*, pages 2190–2194, 1992.

[76] P.M. Young, M.P. Newlin, and J.C. Doyle. Let's get real. In *IMA Proceedings of Robust Control Theory*, volume **66**, pages 143–173, 1995.

[77] G. Zames. On the input-output stability of nonlinear time-varying feedback systems, parts i and ii. In *IEEE Transactions on Automatic Control*, volume **11**, pages 228–238 and 465–476, 1966.

[78] K. Zhou, K. Glove, and J.C. Doyle. *Robust and Optimal Control.* Prentice Hall, New Jersey, 1995.

[79] X. Zhu. Probabilistic μ upper bound using linear cuts. In *Proceedings of the 14th IFAC World Congress*, pages 389–394, 1999.

[80] X. Zhu. Improved bounds computation for probabilistic μ. In *Proceedings of the American Control Conference*, 2000. To appear.

[81] X. Zhu, Y. Huang, and J.C. Doyle. Soft vs. hard bounds in probabilistic robustness analysis. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3412–3417, 1996.

[82] X. Zhu, S. Khatri, and P. Parrilo. μ with linear cuts: Upper bound computation. In *Proceedings of the American Control Conference*, pages 2370–2374, 1999.