To the Graduate Council:

I am submitting herewith a dissertation written by Gwendolyn Harville Walton entitled "Generating Transition Probabilities for Markov Chain Usage Models". I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

J. H. Poore, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor and
Dean of the Graduate School

GENERATING TRANSITION PROBABILITIES FOR MARKOV CHAIN USAGE MODELS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Gwendolyn Harville Walton

May 1995

Copyright © 1995 by Gwendolyn Harville Walton

All rights reserved

Dedication

To Bob, Jennifer, Julia, and Brian

Acknowledgments

I thank Dr. Jesse Poore for the opportunity to participate in this research and Dr. Poore and Dr. Carmen Trammell for the opportunity to work in the University of Tennessee's Software Quality Research Laboratory. I thank Dr. Ken Gilbert, Dr. Mark Jones, and Dr. Michael Thomason for their helpful suggestions and their careful review of this dissertation.

I appreciate the suggestions and encouragement I have received from my colleagues at the Software Quality Research Laboratory and Software Engineering Technology, Inc.

Most of all I thank my family for their love and support.

Abstract

In statistical testing of software, a software usage model is developed to characterize a population of uses of the software. The model is used to plan a testing program and later to generate a statistically correct sample of test cases (uses of the software). Performance on the sample is used as a basis for generalizations about operational reliability.

Although the usage model is developed from the software specifications, typically there is insufficient information to completely specify all one-step transition probabilities in the Markov chain. This work applies techniques from mathematical analysis, mathematical programming, linear algebra, and information theory to present a new approach to the representation and optimization of the transition probabilities of software usage models. New contributions are:

- The application of mathematical constraints and objective functions to manage information about expected software use and test management goals.
- The development of an iterative process using convex programming to generate Markov chain transition probabilities that satisfy all known constraints and optimize an objective function.
- The description and demonstration of some standard, useful constraints and objective functions to support statistical testing.
- The development of a new specification complexity metric.

Table of Contents

1 Introduction 1
1.1 Statistical Testing of Software Based on a Usage Model . . . 1
1.2 Overview and Outline . . . 5

2 Mathematical Principles 7
2.1 Convex Programming . . . 7
2.2 Markov Chains . . . 10
2.3 Entropy . . . 12

3 Generating Transition Probabilities 16
3.1 Constraints and Objectives . . . 16
3.2 Generating Transition Probabilities . . . 18

4 Measuring Complexity and Coverage of Software Specifications 24
4.1 State and Arc Coverage . . . 24
4.2 Specification Complexity and Path Coverage . . . 30
4.2.1 Canonical Paths . . . 30
4.2.2 Calculating Specification Complexity . . . 35
4.2.3 Interpretation and Illustration . . . 37
4.3 Optimization . . . 41
4.3.1 Theorems Concerning Constraints . . . 44
4.3.2 Formulation of Convex Programming Problems . . . 47

5 Application of the Maximum Entropy Principle 52
5.1 Weighted Mean State Entropy and Source Entropy . . . 52
5.2 Stationary Distribution Entropy . . . 54
5.3 Test Trajectory Entropy . . . 54
5.4 Illustration . . . 55
5.5 Recommended Process . . . 62

6 Conclusion 64
6.1 Summary . . . 64
6.2 Recommendations for future work . . . 66

Bibliography 68

Appendix 73

Vita 80

List of Tables

1 Progression of Models 21

2 Transition Matrix for 15-State Usage Model 26

3 State Statistics 27

4 Arc Coverage Statistics 29

5 Trajectory Entropy 39

6 Test Trajectory Entropy, Entire Model 40

7 Test Trajectory Entropy, Upper Section 42

8 Test Trajectory Entropy, Lower Section 43

9 Progression of P for Weighted Mean State Entropy 57

10 Progression of π and Entropy Statistics for Weighted Mean State Entropy 58

11 Progression of P for Stationary Distribution Entropy 59

12 Progression of π and Entropy Statistics for Stationary Distribution Entropy 60

13 Summary of Objective Functions and Constraint Functions 65

14 Source Entropy and Derivatives 76

Chapter 1

Introduction

1.1 Statistical Testing of Software Based on a Usage Model

In statistical testing of software, a software usage model is developed to characterize a population of uses of the software. The model is used to plan a testing program and later to generate a statistically correct sample of test cases (uses of the software). Performance on the sample is used as a basis for generalizations about operational reliability.

Research and practice justify model-based statistical usage testing [26]. Duran and Ntafos [7] indicate that random testing can be superior to partition and coverage testing, is more cost effective, and allows the tester to make 'sound reliability estimates.' Hamlet and Taylor [12] provide further support for the use of random testing. They conclude that partition testing should not be used to make inferences about reliability of the software in use.

Currit, Dyers, and Mills [6] describe a method for embedding the software development and testing process within a formal statistical design. Mills [18] points out that testing with selected test cases can provide nothing but anecdotal evidence; statistical testing is needed to scientifically certify the reliability of software. Adams [1] demonstrated that a relatively small proportion of the errors in software account for the vast majority of reported failures in the field. Further, Adams demonstrated that these frequently reported failures occur early in the operational life of the software. Thus, statistical testing based on

a usage model tends to find early in the test cycle those errors that will occur most frequently in operational use. Musa [19] reports a benefit-to-cost ratio of 10 or greater in developing and using operational usage models. Research and practice support the argument that, "apart from certain anecdotal tests that are justified on legal, moral, ethical, or political grounds, the most cost effective way to spend the software test budget is on statistical testing based on a usage model." [26]

Whittaker, Thomason, and Poore [27, 29, 30] demonstrated the suitability of treating software testing as a stochastic process and modeling usage by finite state, discrete parameter, time homogeneous, irreducible Markov chains. The structure of a Markov chain usage model represents the specification of what the software is capable of doing. The state transition probabilities represent an environment of use or class of users. When the Markov chain is represented as a stochastic matrix P, the value $p_{i,j}$ in each cell is the probability of a transition from state s_i to state s_j. Thus, the zero entries represent the impossible transitions (absence of arcs in the graphical rendering), the ones represent the certain transitions (single exit arcs), and all other cells contain transition probabilities $0 < p_{i,j} < 1$ (states with multiple exit arcs).

Techniques to analyze the usage model that provide insight into the testing process are detailed in [27, 29, 30].

The key steps in developing a Markov chain usage model are presented in [26]:

- Develop model structure based on the specification of the software.
- Assign exit arc transition probabilities for the model based on any information available concerning intended use of the software and test objectives.

- Compute usage statistics for the chain.
- Review the model and usage statistics with respect to the software specification and the testing plan.
- Revise the model as required to make it more realistic with respect to the application.

Since the usage model is based on functional and usage specifications, all the information required to develop the model should be available before a line of code is written. This allows total independence between the software development team and the usage modeling and test team. It allows test planning to occur in parallel with the software development, thus shortening the elapsed time required to develop and deliver software. The model provides information to developers concerning the importance (in terms of probability of use or in terms of testing objectives) of each feature, provides the information required by testers to construct effective and informative test cases and, with statistical testing, provides a basis from which inferences of software reliability may be made.

Experience in statistical testing of software and in building, analyzing, and applying Markov chain usage models is accumulating. Projects have been completed successfully in several companies and government agencies [2, 28]. The environments range from real-time, embedded code on a bare machine to sophisticated software engineering environments.

Although the usage model is developed from the software specifications, typically there is insufficient information to completely specify all one-step transition probabilities

in the Markov chain. Three approaches to defining transition probabilities have been used [27, 29]:

- The "informed model", where transition probabilities are assigned based on field data.
- The "intended model", where transition probabilities are assigned based on stipulations (informed assumptions) about expected use.
- The "uninformed model", where, because no information is available concerning expected use of the software, uniform probabilities are assigned across the exit arcs for each state of the Markov chain.

Because the availability of information concerning expected use of the software typically varies across the model, the current practice is to use a combination of these three approaches to define and update the transition probability matrix by assignment of specific values to individual transition probabilities. Using this cycle of assigning, verifying, and revising transition probabilities can be difficult for a variety of reasons:

- Typically, there is insufficient information to completely specify all probabilities.
- The test objectives and constraints often are functions of transition probabilities of multiple states.
- The transition probabilities must be updated as new information concerning expected use of the software becomes available or as testing objectives change.
- A change to the exit arc transition probabilities of a single state may impact the frequency of occurrence within test cases of every other state.

- Managing changes can be difficult because of the need to identify and preserve implicit relationships among probability values.

Better practices are needed for establishing transition probabilities for Markov chain usage models. This research provides them.

1.2 Overview and Outline

This work applies techniques from mathematical analysis, mathematical programming, linear algebra, and information theory to present a new approach to the representation of usage models and the optimization of the transition probabilities relative to testing objectives. New contributions are:

- The application of mathematical constraints and objective functions to manage information about expected software use and test management goals.
- The development of an iterative process using convex programming to generate Markov chain transition probabilities that satisfy all known constraints and optimize an objective function.
- The description and demonstration of some standard, useful constraints and objective functions to support statistical testing.
- The development of a new specification complexity metric.

Chapter 2 presents a brief overview of the mathematical principles necessary for the theorems and interpretations in this research. The chapter contains no new results; rather it is a compilation of background material from several sources in the fields of mathematical analysis, mathematical programming, and information theory.

Chapter 3 illustrates the convex programming approach to the generation of Markov chain transition probabilities. Procedures are presented for describing basic relationships on single-state exit arcs, among exit arcs of more than one state, and sub-models. A recommended iterative optimization process for model development is described. The information in this chapter is also included in Poore, Walton, and Whittaker [22].

Chapter 4 examines test coverage measures, i.e., prospective coverage of the usage model during testing. A new measure of specification complexity based on the statistically typical paths through the model of the specification is derived. Relevant software engineering issues are considered with respect to each form of coverage in usage model optimization (state, arc, path). The behavior of the associated functions are characterized, and recommendations are given for formulation of objective functions and constraints. The information in this chapter is included in Walton and Poore [25].

Chapter 5 describes the applicability of the maximum entropy principle to usage model optimization. A recommended procedure is described for the use of convex programming and the maximum entropy principle to optimize the assignment of Markov chain usage model transition probabilities based on specific testing goals, represented in objective functions and constraints.

Chapter 6 summarizes these results and provides a list of the constraint functions and objective functions that have been derived to generate transition probabilities through usage model optimization. The direction of continued research in the application of operations research to software testing is given.

Chapter 2

Mathematical Principles

This chapter presents an overview of the background material in the fields of mathematical analysis, mathematical programming, and information theory necessary for the theorems and applications in this research.

2.1 Convex Programming

Usage model optimization requires a technique for determining the values of a finite set of Markov chain transition probabilities which optimize (maximize or minimize) an objective function subject to a finite set of mathematical constraints. The general problem of optimizing any function subject to a set of unrestricted constraints can be analytically or computationally intractable. The problem is tractable when it is restricted to convex programming, the minimization of a convex objective function subject to a finite set of convex constraints. The best method for a particular problem will depend on properties of the objective function and constraint functions (linear, quadratic, strictly convex, convex, differentiable, continuous), size of the problem, required accuracy of the solution, and availability of software to support particular optimization methods. The challenge in using convex programming is in achieving, maintaining, and restoring convexity in all the equations while modeling the real-world problem. The challenge in the application of convex programming to Markov chain usage models is to define convex functions on the transition probabilities that meaningfully constrain the stochastic behavior of the usage model.

There are well-defined computational techniques for minimizing a convex, continuously differentiable objective function (or maximizing a concave, continuously differentiable objective function) over a non-empty, closed, and convex feasible region [5, 10]. These techniques are suitable for the purposes of defining the structure and determining the parameters of software usage models.

One technique for determining the convexity of a function is to examine the Hessian. If the function is twice continuously differentiable over the region of interest, the Hessian will be nonnegative definite if and only if the function is convex; and the Hessian will be positive definite if and only if the function is strictly convex.

When a strictly convex objective function is minimized subject to a set of constraints that define a closed convex feasible region, the optimal solution is a unique global minimum [5,10]. However, if the objective function is convex, but not strictly convex (for example, a linear objective function), the optimal solution is a global minimum, but not necessarily a unique global minimum. If F(x) is a (strictly) convex function, then $(-F(x))$ is a (strictly) concave function. Thus, when a (strictly) concave objective function is maximized subject to a set of constraints that define a closed convex feasible region, the solution is a (unique) global maximum.

The basic formulation for a convex programming problem is to find values for decision variables $\{x_1, x_2, \ldots, x_n\}$ which minimize a convex objective function (or maximize a concave objective function) $Y = f(x_1, x_2, \ldots, x_n)$ subject to a finite set of convex constraints:

$$g_1(x_1, x_2, \ldots, x_n) = 0$$

$$g_2(x_1, x_2, \ldots, x_n) = 0$$

. . .

$$g_m(x_1, x_2, \ldots, x_n) = 0$$

The constrained optimization problem to minimize $Y(x)$, $x = (x_1, x_2, \ldots, x_n)$, subject to $g_i(x) = 0$, $i = 1, \ldots, m$ is equivalent to minimizing an unconstrained Lagrangian function:

$Z(x, v) = Y(x) + \sum_{i=1}^{m} v_i g_i(x)$, where the v_i are Lagrange multipliers.

If the Lagrangian is convex, it can be solved by any unconstrained convex optimization solver. If all the functions are twice continuously differentiable over the region of interest, the Lagrangian can be solved by Newton's algorithm [10] using an initial feasible vector x and an initial positive vector v. Newton's algorithm will yield a candidate for the global maximum or minimum.

Usage model optimization may involve solving linear systems when some or all the coefficients of the system are not sharply defined (i.e., when some or all of the coefficients of the system are defined on specified intervals rather than defined as specific values.) Linear programming procedures can be used to: (1) determine if there is at least one feasible solution for such a linear system, and (2), if there is a feasible solution, determine when the boundaries of the intervals are "binding" and when they are "slack". These procedures are described by Oettli [20], Oettli, Prager, and Wilkinson [21], and Smith [24].

The material in this section is available in many mathematical analysis and optimization texts. For more information on convex functions or convex programming, the reader is referred to [3, 5, 10].

2.2 Markov Chains

Markov chain usage models [27, 29, 30] are finite state, discrete parameter, time homogeneous, irreducible Markov chains. The usage Markov chain has a unique start state s_{Inv} (that represents invocation of the software), a unique final state s_{Term} (that represents termination of the software), a set of intermediate usage states, and transition arcs between states as defined by the software specification. The Markov property requires that the next state is independent of all past states given the present state. The Markov chain can be represented as a two dimensional row-stochastic, irreducible matrix P. For each i = 1,...,n, row i of the transition matrix P for an n-state usage model is the probability distribution associated with transitions from state s_i. Thus, $p_{i,j}$ is the probability that the next state will be s_j given that current state is s_i.

Since a usage model is an irreducible Markov chain, the stationary distribution vector $\pi = \pi P$ exists and is unique [16]. Each row of $P^{\infty} = \lim_{n \to \infty} P^n$ is equal to the vector π. The i-th element of π is the long run average probability of the occurrence of state s_i.

Any computation involving an element of π represents long run behavior. When the amount of planned testing is sufficient to yield a set of test cases that is statistically

representative of the usage model, long-run behavior is applicable. It is important to note that any change to the transition matrix may change its stationary distribution vector.

Consider transition matrix P for an n-state usage model with invocation state s_{Inv} and termination state s_{Term} such that each element of P is defined on a known, closed interval $0 \le \alpha_{i,j} \le p_{i,j} \le \beta_{i,j} \le 1$. The probability that state s_j occurs in a test case is equal to the probability $y_{inv,j}$ that the process starting in state s_{Inv} ends in state s_j, given that states s_j and s_{Term} are absorbing states [27]. If Q is the absorbing Markov chain derived from P with absorbing states s_{Term} and s_j, then $y_{i,j}$ can be calculated by [16]:

$$y_{i,j} = q_{i,j} + \sum_{k \notin \{j, Term\}} q_{i,k} y_{k,j} \text{ for } i, j \in \{1, \ldots n\} .$$

The following facts have been established [27, 28]:

- The expected number of sequences (test cases) that will be required before a particular state s_j occurs is $(1/y_{Inv,j})$
- The expected number of test cases required to cover the least likely state is the maximum of the $(1/y_{Inv,j})$ over all states s_j
- The expected test case length, or the expected number of state transitions in a single test case, is $((1/\pi_{inv}) - 1)$
- The expected number of sequence changes, in the long run, between occurrences of state s_j is (π_{Inv}/π_j)
- The expected number of occurrences of state s_j per test case is (π_j/π_{Inv})
- The expected probability of occurrence of state s_j during all testing is π_j
- The expected number of test cases that will be required before arc (j,k) occurs is

$(1/(y_{Inv,j}\ p_{j,k}))$

- The expected number of test cases that will be required to cover the least likely arc in the usage model is equal to the maximum of the $(1/(y_{Inv,j}\ p_{j,k}))$ for all j
- The expected number of occurrences of arc (j,k) per test case is $((\pi_j\ p_{j,k})/\pi_{Inv})$
- The expected probability of occurrence of arc (j,k) during all testing is $(\pi_j\ p_{j,k})$

Meyer [17] proved the following concerning group inverses of the transition matrices of finite state irreducible Markov chains:

- A finite state, irreducible Markov chain with transition probability matrix P has the property that $A = I - P$ belongs to a multiplicative matrix group, where I is the identity matrix.
- The group inverse of A is unique, guaranteed to exist, and satisfies $AA^{\#}A = A$, $A^{\#}AA^{\#} = A^{\#}$, and $AA^{\#} = A^{\#}A$
- The group inverse is applicable for generating usage statistics for Markov chains.

For more information on Markov chains, the reader is referred to Kemeny and Snell [16] and Meyer [17]. For more information on the application of Markov chains to software usage models, the reader is referred to Whittaker [27], Whittaker and Poore [29], and Whittaker and Thomason [30].

2.3 Entropy

Entropy of a random variable X, denoted H(X), is a measure of the average uncertainty associated with the events $X = \{x_i\}$ for i = 1, ..., n. The greater the entropy, the more uncertain the outcome or behavior. Another interpretation of entropy is the mini-

mum average number of "yes or no" questions required to determine the result of one observation of X [23].

Entropy of the probability distribution associated with random variable X is defined by [23] as:

$$H(X) = -\sum_{i=1}^{n} \text{prob}(X = x_i) \log(\text{prob}(X = x_i))$$

The choice of logarithmic base defines the unit of measure. Base 2 is used in this paper.

For two random variables X and Y (not necessarily independent), the conditional entropy of Y given X is the average of the entropy of Y for each value of X, weighted according to the probability of occurrence of that value of X:

$$H(Y|X) = \sum_{i} \text{prob}(X = x_i) H(Y|X = x_i)$$

Thus the joint entropy of two independent probability distributions is equal to the sum of the entropy of each probability distribution.

Theorem 2.1 H(X) is a strictly concave function.

Proof:

Let $p_i = \text{prob}(X = x_i)$.

Then: $H(X) = -\sum_{i} p_i \log(p_i)$

$$\log(p_i) = \frac{1}{\ln(2)} (\ln(p_i))$$

$$\frac{\partial}{\partial x_i} H(x) = -\left(\frac{1}{\ln(2)}\right)(\log(p_i) + 1)$$

The Hessian is:

$$\frac{1}{\ln(2)}\begin{bmatrix} -1/p_1 & & & \\ & -1/p_2 & 0 & \\ & & \cdots & \\ & 0 & & \cdots \\ & & & -1/p_n \end{bmatrix}$$

The Hessian is negative definite; therefore H(X) is strictly concave. ❐

Theorem 2.2 Joint entropy of two random variables X and Y (not necessarily independent), denoted H(Y|X), is a strictly concave function if the probability distribution for *X* is fixed.

Proof: For each constant value of x_i, the entropy of the random variable $Y|X = x_i$ is a strictly concave function. The weighted sum of strictly concave functions is strictly concave when all weights are fixed and non-negative. Therefore H(Y|X) is strictly concave. ❐

Source entropy of a Markov chain M having transition matrix P, denoted H{M}, is the average uncertainty per step and is defined by [4]:

$$H\{M\} = -\sum_i \pi_i \sum_j p_{i,j} \log(p_{i,j})$$

Source entropy is not a concave function. See the appendix for a discussion of this issue.

Theorem 2.3 Given a Markov chain M represented by transition matrix P with a fixed stationary probability distribution π, H{M} is a strictly concave function.

Proof: For fixed π, H{M} is the weighted sum of strictly concave functions. Therefore H{M} is strictly concave. ❐

Scientists in many fields have accepted the maximum entropy principle as the basis for selecting a probability distribution when only partial information is available about the random variable [15]. Jaynes [13] describes the maximum entropy estimate as: "the least biased estimate possible on the given information; i.e., it is maximally noncommittal with regard to missing information". Thus, according to the maximum entropy principle, when multiple values of transition probabilities satisfy all the known constraints for the usage model, the transition probability matrix should be selected based on maximizing the entropy of one or more of the probability distributions associated with the Markov chain usage model.

For more information on entropy, the reader is referred to Shannon [23] and Ash [4]. For more information on the maximum entropy principle, the reader is referred to Jaynes [13] and Kapur [15].

Chapter 3

Generating Transition Probabilities

The current practice for developing Markov chain usage models is to define and update transition probabilities by assignment of specific values to individual transition probabilities. This chapter demonstrates an improved approach that uses convex programming to generate Markov chain transition probabilities based on constraints and objective functions. The approach allows for better documentation of the relationships between the various transition probabilities and better control of the usage modeling and testing processes. Procedures are presented for describing basic relationships on single-state exit arcs, and among exit arcs of more than one state and sub-models. A recommended iterative optimization process for model development is described.

3.1 Constraints and Objectives

A Markov chain of n states represented by a transition matrix P will have n^2 transition probabilities, $\{p_{1,1}, \ldots, p_{n,n}\}$. Of the n^2 probabilities, only those with values not equal to 0 or 1 need to be determined. The convex problem to be solved will have a decision variable for each transition probability not equal to 0 or 1. Usage models tend to be very sparse, some having fewer than one percent non-zero cells for large models. Thus, the set of decision variables $\{x_1, \ldots, x_m\}$ is a relatively small subset of the transition probabilities. Even large usage models lead to convex programming models that are considered small in the operations research community.

Constraints arise from three sources:

- Structural constraints that define whether a transition is allowed between two states. These constraints can be represented by linear functions: (1) certain $p_{i,j} = 0$; (2) certain $p_{i,j} = 1$; and (3) $\sum_{j=1,n} p_{i,j} = 1$ for $i = 1, ..., n$.
- Usage constraints that define values for particular transition probabilities or relationships between sets of transition probabilities based on expected usage of the software.
- Management constraints that define testing objectives and that define the cost and value of testing particular sections of the usage model.

Any intended use or test constraint that is best expressed at the level of individual transition probabilities can be made by a constraint of the form $p_{i,j} = c$. If the level of knowledge is a relationship, e.g., that the probability on the exit arc $p_{i,x}$ should be twice that on exit arc $p_{i,y}$, but a more specific description is not available, then the constraint should be $p_{i,x} - 2p_{i,y} = 0$, and the specific values for $p_{i,x}$ and $p_{i,y}$ should be determined in consideration of all other constraints. Even models based on extensive field data or detailed test plans, with many probability constants known, usually have many other probabilities that must be determined without data. Since there can be many sets of transition probabilities that collectively satisfy a system of high-level constraints, test objectives must be used to select among the choices.

Standard techniques of mathematical analysis can be used to analyze the constraint and objective functions. The problem of optimizing Markov chain transition probabilities becomes a mathematical programming problem concerning the identification of a global

maximum or minimum for the objective function satisfying the constraints when evaluated over a closed set of Markov chain transition probability matrices.

3.2 Generating Transition Probabilities

Figure 1 gives an overview of a recommended process for generating an optimal usage model. The usage model optimization process accepts as input the usage model structure, expected usage information, and test management policies and generates as output the optimal transition probability matrix for the model. The first step is to use only the usage model structure constraints and define the transition probabilities by maximizing a baseline objective function. A usage model analysis is then performed on the initial model.

An iterative cycle follows: (1) modify the objective function based on test management policies and the usage model analysis; (2) incrementally add constraints based on expected usage information, test management policies, and usage analysis results; (3) generate the optimal transition probability matrix for the new model; and (4) analyze the new model to assess the impact of the refinements. The process continues until usage model analysis yields satisfactory results.

Consider, for example, the usage model with transition probability matrix P as graphically depicted in Figure 2. Table 1 gives key results of the process of building on a baseline model constraint-by-constraint to assign probabilities for the arcs from states s_2 and s_9. The structure of this model leads to a convex programming model with a set of 10 decision variables:

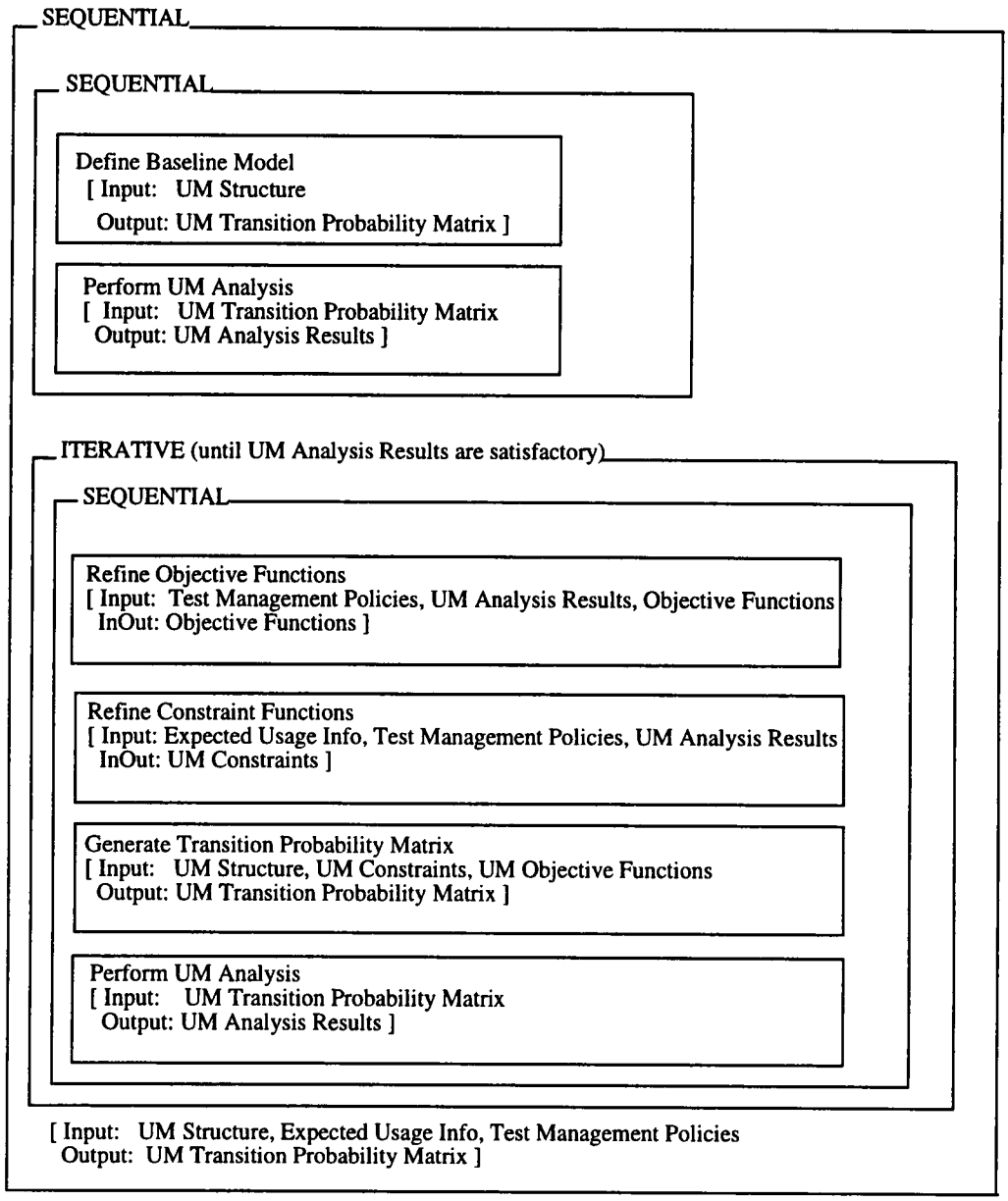


Figure 1: Usage Model Optimization Process

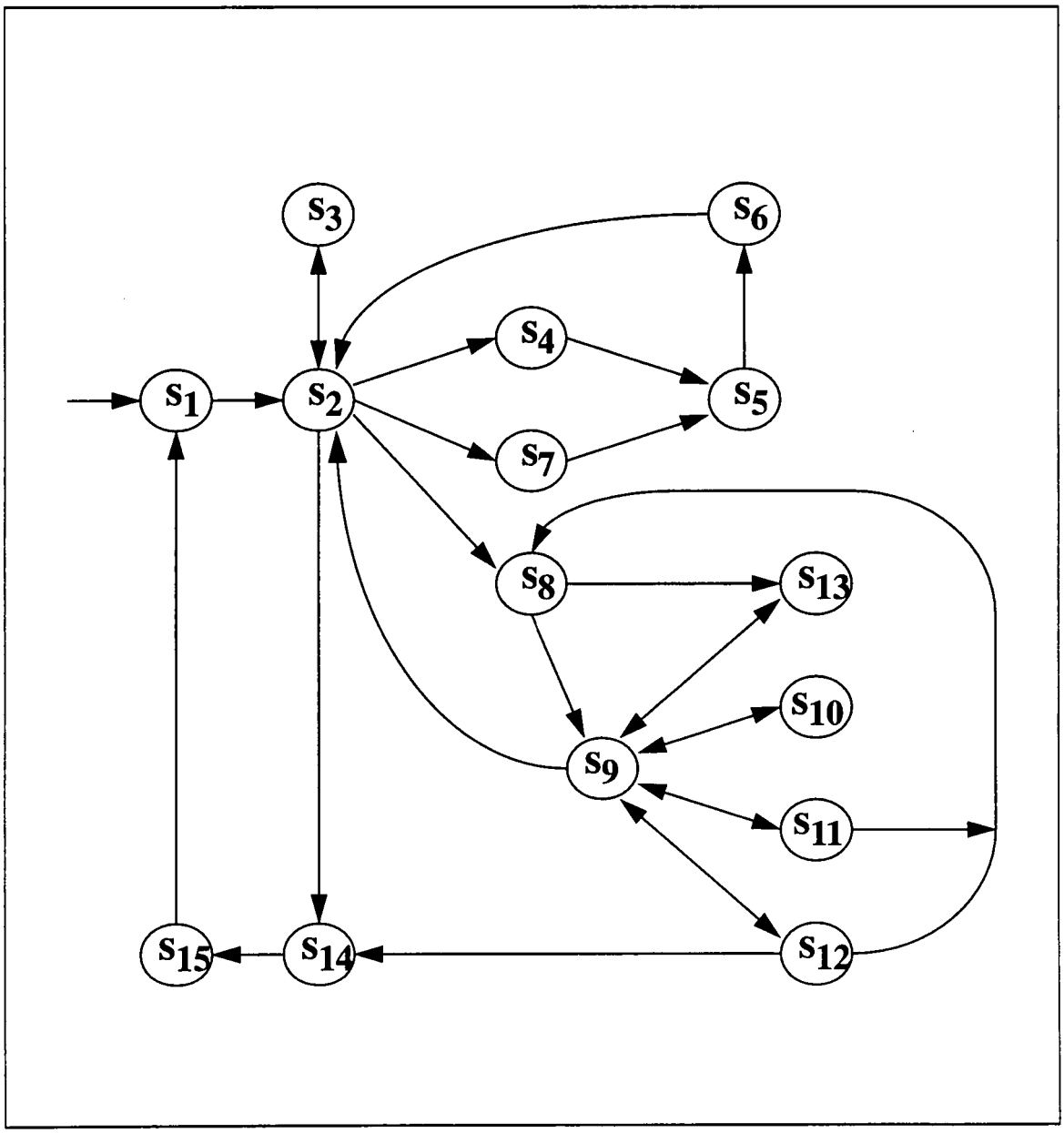


Figure 2: Graphical Representation of a 15-State Usage Model

Table 1: Progression of Models

	g_1&g_2	& g_3	& g_4	& g_5	& g_6	& g_7	& g_8	& g_9	& g_{10}
$p_{2,3}$	0.200	0.210	0.210	0.231	0.231	0.221	0.083	0.083	0.083
$p_{2,4}$	0.200	0.210	0.210	0.231	0.231	0.221	0.083	0.083	0.083
$p_{2,7}$	0.200	0.210	0.210	0.133	0.133	0.112	0.167	0.167	0.167
$p_{2,8}$	0.200	0.092	0.092	0.101	0.101	0.112	0.167	0.167	0.167
$p_{2,14}$	0.200	0.277	0.277	0.304	0.304	0.335	0.500	0.500	0.500
$p_{9,2}$	0.200	0.200	0.284	0.266	0.266	0.281	0.241	0.267	0.267
$p_{9,10}$	0.200	0.200	0.071	0.066	0.066	0.070	0.060	0.067	0.067
$p_{9,11}$	0.200	0.200	0.215	0.266	0.266	0.224	0.333	0.333	0.333
$p_{9,12}$	0.200	0.200	0.215	0.201	0.201	0.213	0.183	0.167	0.200
$p_{9,13}$	0.200	0.200	0.215	0.201	0.201	0.213	0.183	0.167	0.133
v_1	0.609	0.559	0.559	0.466	0.465	0.512	0.158	0.063	-0.167
v_2	0.609	0.609	0.537	0.603	0.603	0.548	0.700	0.792	1.015
v_3		-0.275	-0.275	-0.275	-0.275	-0.420	-0.464	-0.370	-0.140
v_4			-0.277	-0.277	-0.277	-0.277	-0.277	-0.470	-0.693
v_5				-0.277	-0.277	-0.050	-0.601	-0.886	-1.576
v_6					0.000	0.000	-1.327	-1.422	-1.652
v_7						0.580	0.759	0.380	-0.540
v_8							-1.327	-1.422	-1.652
v_9								0.193	0.659
v_{10}									-0.406

$$\{p_{2,3}, p_{2,4}, p_{2,7}, p_{2,8}, p_{2,14}, p_{9,2}, p_{9,10}, p_{9,11}, p_{9,12}, p_{9,13}\}$$

The progression of transition probabilities presented in Table 1 was obtained from the following unconstrained Lagrange function built from a strictly convex objective function, Y(P), and ten constraint functions, $g_i(P)$, and their corresponding Lagrange multipliers, v_i:

Minimize $Z = Y(P) + \sum_{i=1,10} v_i g_i(P)$, for

$Y(P) = (-(1/2))\,(H(s_2) + H(s_9))$ where $H(s_i)$ is the entropy of state s_i

$g_1(P) = p_{2,3} + p_{2,4} + p_{2,7} + p_{2,8} + p_{2,14} - 1 = 0$

$g_2(P) = p_{9,2} + p_{9,10} + p_{9,11} + p_{9,12} + p_{9,13} - 1 = 0$

$g_3(P) = p_{2,14} - 3p_{2,8} = 0$

$g_4(P) = p_{9,2} - 4p_{9,10} = 0$

$g_5(P) = p_{9,11} - 2p_{2,7} = 0$

$g_6(P) = p_{2,3} - p_{2,4} = 0$

$g_7(P) = p_{2,7} - p_{2,8} = 0$

$g_8(P) = p_{2,7} - 2p_{2,3} = 0$

$g_9(P) = p_{9,11} - 5p_{9,10} = 0$

$g_{10}(P) = p_{9,12} - 3p_{9,10} = 0$

Note that the first two constraints (indicated on Table 1 as "g_1&g_2") are the structural constraints associated with states s_2 and s_9. The next two constraints, g_3 and g_4, are single-state constraints. Constraint g_5 is the first constraint that establishes a dependency between states s_2 and s_9. Since all the functions are twice continuously differentiable over

the region of interest (i.e., over the open interval (0,1)), the Lagrangian can be solved by Newton's algorithm [10].

Note that $v_6 = 0$ when g_6 first appears. This indicates that g_6 contributed no new information in this step of the optimization process. However, as the change in v_6 shows, the addition of g_8 caused g_6 to become meaningful. Such facts are difficult to discern by examining the hundreds of transition probabilities that might be present in the usage model of a real product, and are typical of the insight that can be gained from post-optimality analysis.

Chapter 4

Measuring Complexity and Coverage of Software Specifications

Coverage testing in the context of Markov chain usage models refers to coverage of a model of the specification and profile of intended use, rather than coverage of the code that implements the specification. Test cases are sequences randomly generated from the usage model. Each usage model can be analyzed to determine the expected extent of testing necessary to achieve coverage of all states, coverage of all arcs, and coverage of each unique path within the set of statistically typical paths (called canonical paths). A new measure of specification complexity based on the number of canonical paths through the model of the specification is derived. A formula is presented to compare across different usage models the amount of testing required to achieve coverage of the canonical paths. Convexity properties are established for this formula and for formulas to compute the expected number of test cases required to achieve state coverage and arc coverage. The formulas are studied for use in optimization problems to generate transition probabilities.

4.1 State and Arc Coverage

In the context of statistical testing based on Markov chain usage models, achieving state coverage means that each state in the usage model occurs at least once in the randomly generated set of test cases.

Consider transition matrix P where each element of P is defined on a known, closed interval $0 \leq \alpha_{i,j} \leq p_{i,j} \leq \beta_{i,j} \leq 1$. For the 15-state usage model of Figure 2, consider two versions for the transition matrix: Version 1, uniform transition probabilities for each state and Version 2, non-uniform transition probabilities for states s_2 and s_{11}. For each version, Table 2 gives P, and Table 3 gives π_j, $y_{Inv,j}$, and $(1/y_{Inv,j})$. Table 3 shows that 3.667 test cases are expected to be required to cover the least likely state for the uniform probabilities of Version 1, and 8.333 test cases are expected to be required to cover the least likely state if states s_2 and s_{11} have the non-uniform transition probabilities of Version 2.

The expected amount of testing (total number of state transitions) for coverage of a particular state is a function of both the expected test case length and the expected number of test cases required to cover the state. Reducing the expected number of test cases required to cover a particular state may increase the test case length and will not necessarily reduce the amount of testing required to cover the state. Thus, to reduce the total number of state transitions before a particular state s_j occurs requires that we reduce $(((1/\pi_{Inv}) - 1)(1/y_{Inv,j}))$.

The expected number of test cases required to cover all states of the model is bounded below by the maximum of the $1/y_{Inv,j}$ over all states s_j and is bounded above by $\sum_i 1/y_{Inv,i}$ for the states s_i that cannot be eliminated by structural analysis. The upper bound is not tight. To reduce the total number of state transitions before state coverage occurs requires that we reduce $((1/\pi_{Inv}) - 1) \sum_i (1/y_{Inv,i})$.

Table 2: Transition Matrix for 15-State Usage Model

Version 1: Uniform Probabilities

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_1	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
s_2	[illegible]														
s_3	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
s_4	0.	0.	0.	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
s_5	0.	0.	0.	0.	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.
s_6	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
s_7	0.	0.	0.	0.	1.00	0.	0.	0.	0	0.	0.	0.	0.	0.	0.
s_8	0.	0.	0.	0.	0.	0.	0.	0.	0.50	0.	0.	0.	0.50	0.	0.
s_9	0.	0.20	0.	0.	0.	0.	0.	0.	0.	0.20	0.20	0.20	0.20	0.	0.
s_{10}	0.	0.	0.	0.	0.	0.	0.	0.	1.00	0.	0.	0.	0.	0.	0.
s_{11}	[illegible]														
s_{12}	0.	0.	0.	0.	0.	0.	0.	0.33	0.33	0.	0.	0.	0.	0.33	0.
s_{13}	0.	0.	0.	0.	0.	0.	0.	0.	1.0	0.	0.	0.	0.	0.	0.
s_{14}	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.00
s_{15}	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.

Version 2: Non-Uniform Probabilities for States s2 and s11

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
s1	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
s2	[illegible]														
s3	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
s4	0.	0.	0.	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
s5	0.	0.	0.	0.	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.
s6	0.	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
s7	0.	0.	0.	0.	1.00	0.	0.	0.	0	0.	0.	0.	0.	0.	0.
s8	0.	0.	0.	0.	0.	0.	0.	0.	0.50	0.	0.	0.	0.50	0.	0.
s9	0.	0.20	0.	0.	0.	0.	0.	0.	0.	0.20	0.20	0.20	0.20	0.	0.
s10	0.	0.	0.	0.	0.	0.	0.	0.	1.00	0.	0.	0.	0.	0.	0.
s11	[illegible]														
s12	0.	0.	0.	0.	0.	0.	0.	0.33	0.33	0.	0.	0.	0.	0.33	0.
s13	0.	0.	0.	0.	0.	0.	0.	0.	1.0	0.	0.	0.	0.	0.	0.
s14	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.00
s15	1.00	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.

Table 3: State Statistics

Version 1: Uniform Probabilities

	π	$y_{Inv,j}$	$1/(y_{Inv,j})$
s_1	0.050	1.000	1.000
s_2	0.201	1.000	1.000
s_3	0.040	0.444	2.250
s_4	0.040	0.444	2.250
s_5	0.080	0.615	1.625
s_6	0.080	0.615	1.625
s_7	0.040	0.444	2.250
s_8	0.065	0.500	2.000
s_9	0.150	0.500	2.000
s_{10}	0.030	0.273	3.667
s_{11}	0.030	0.273	3.667
s_{12}	0.030	0.333	3.000
s_{13}	0.063	0.417	2.400
s_{14}	0.050	1.000	1.000
s_{15}	0.050	1.000	1.000

Version 2: Non-Uniform Probabilities for States s_2 and s_{11}

	π	$y_{Inv,j}$	$1/(y_{Inv,j})$
s_1	0.072	1.000	1.000
s_2	0.239	1.000	1.000
s_3	0.050	0.412	2.426
s_4	0.050	0.412	2.426
s_5	0.100	0.584	1.713
s_6	0.100	0.584	1.713
s_7	0.050	0.412	2.426
s_8	0.031	0.250	4.000
s_9	0.083	0.250	4.000
s_{10}	0.017	0.120	8.333
s_{11}	0.017	0.120	8.333
s_{12}	0.017	0.143	7.000
s_{13}	0.032	0.195	5.121
s_{14}	0.072	1.000	1.000
s_{15}	0.072	1.000	1.000

In the context of statistical testing based on Markov chain usage models, achieving arc coverage means that each arc in the usage model is traversed at least once in a random set of test cases. For the 15-state usage model of Figure 2, Table 4 shows that 10.0 test cases are expected to be required to cover the least likely arc when the transition probabilities for each state are uniform. Further, Table 4 shows that the arcs in the set {(9,2), (9,10), (9,11), (9,12), (9,13)} are expected to be the last arcs covered for this version of the model. When two states (s_2 and s_{11}) have the non-uniform transition probabilities listed in the one-step transition matrix in Version 2 of Table 2, Table 4 shows that 41.7 test cases are expected to be required to cover the least likely arc. Further, Table 4 shows that the arc (11,8) is expected to be the last arc covered; and we expect 21.7 more test cases to cover arc (11,8) than to cover any of the arcs in the set {(9,2), (9,10), (9,11), (9,12), (9,13)}.

As was the case for state coverage, the expected amount of testing to cover an arc is a function of both the expected test case length and the expected number of test cases required to cover the arc. To reduce the total amount of testing before a particular arc (j,k) occurs requires that we reduce $((1/\pi_{Inv}) - 1)(1/y_{Inv,j})(1/p_{j,k})$.

The expected number of test cases required to cover all arcs of the model is bounded below by the maximum of the $(1/y_{Inv,j})(1/p_{j,k})$ over all states s_k and all states s_j and is bounded above by $\sum_i (1/y_{Inv,i}) \sum_k (1/p_{i,k})$ for all states s_k and the states s_i that cannot be eliminated by structural analysis. The upper bound is not tight. To reduce the total number of state transitions before arc coverage occurs requires that we reduce $((1/\pi_{Inv}) - 1) \sum_i (1/y_{Inv,i}) \sum_k (1/p_{i,k})$.

Table 4: Arc Coverage Statistics

Version 1: Uniform Probabilities

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_1		1.00													
s_2			5.00	5.00			5.00	5.00						5.00	
s_3		2.25													
s_4					2.25										
s_5						1.63									
s_6		1.63													
s_7					2.25										
s_8									4.00				4.00		
s_9															
s_{10}									3.67						
s_{11}								7.33	7.33						
s_{12}								9.00	9.00					9.00	
s_{13}									2.40						
s_{14}															1.00
s_{15}	1.00														

Version 2: Non-Uniform Probabilities for States s_2 and s_{11}

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_1		1.00													
s_2			4.76	4.76			4.76	10.84						3.61	
s_3		2.43													
s_4					2.43										
s_5						1.71									
s_6		1.71													
s_7					2.43										
s_8									8.00				8.00		
s_9															
s_{10}									8.33						
s_{11}									10.42						
s_{12}								21.00	21.00					21.00	
s_{13}									5.12						
s_{14}															1.00
s_{15}	1.00														

4.2 Specification Complexity and Path Coverage

4.2.1 Canonical Paths

Because the usage model is based on the specification, the model can be developed, verified, and analyzed before code is written. An analysis of the complexity of the model of the specification might lead to simplification of the specification in various ways, before code development begins.

For arbitrary states s_i and s_j, a single *path* from state s_i to state s_j of a Markov chain usage model is a finite sequence of states beginning with state s_i and ending in the first occurrence of state s_j. A Markov chain produces a set of sequences whose ensemble statistics closely match the statistics of the chain. There are $2^{N(H\{M\}+\delta_N)}$ sequences of length N where $\lim_{N\to\infty} \delta_N = 0$ [4]. Thus, for any specified small positive δ there exists N such that we can enumerate a finite set of sequences of length N with total probability of at least $1 - \delta$. In the field of information theory, the ensemble of these sequences is said to be statistically typical. For a Markov chain with state set S, $C \subseteq S$, and states s_i and s_j in C, the statistically typical sequences of the Markov chain yield a finite collection of (not necessarily unique) paths from state s_i to s_j where each state in each path is a member of C. This collection is the statistically typical paths from state s_i to s_j. We call the unique paths in this collection the *canonical paths.*

Analysis of the number of canonical paths from s_i to s_j can provide information that might improve the efficiency of the development and testing processes. The number of canonical paths from state s_i to state s_j where each state in each path is a member of a

specified subset of the model provides a measure of complexity relative to this subset. The number of canonical paths in the model from s_{Inv} to s_{Term} that are allowed to visit any state in the model provides a measure of complexity of the entire model.

If a usage model has any loops other than the one created by the arc from s_{Term} to s_{Inv}, there are infinitely many possible paths from s_{Inv} to s_{Term}. Thus, covering all possible paths with a finite number of test cases is impossible. In treatments of code coverage, the concept of a path is usually defined based on a specific limit on the number of times to traverse a single loop. In a program with several loops, path choices near the end of the program are not related to subpaths traversed earlier. In the context of statistical testing based on Markov chain usage models we assess path coverage with respect to *canonical test paths*, the canonical paths from s_{Inv} to s_{Term}.

A *trajectory* $T\{i, j\}$ is defined in [8] as the set of all paths from state s_i to state s_j with no intervening state s_j. Thus, *trajectory entropy* $H(T\{i, j\})$ provides a measure of the uncertainty in selecting a path from the set $T\{i, j\}$. (More precisely, the entropy is defined on the probability distribution associated with the set of paths.) This definition of trajectory includes paths with s_{Term} as an intervening state (i.e., paths that span multiple test cases). In the context of statistical testing based on test cases generated from a Markov chain usage model, we require a measure of the uncertainty in selecting a path from the set of paths from state s_i to state s_j that have no intervening state s_j or s_{Term} (i.e., where each path occurs entirely within a single execution of the software).

This chapter presents first a generalization on the techniques of [8] and then an adaptation to define such a metric called *test trajectory entropy*, which can be used to mea-

sure specification complexity and perform analyses involving path coverage. To generalize, the *trajectory with respect to a set C* (for C any subset of S, the set of states in the usage model), denoted $TC\{i, j\}$, is the set of all paths from state s_i to state s_j such that each path includes only states from set C, and s_j occurs only once in the path (i.e., s_j is not an intervening state, and $i \neq j$). The set $C\{j\} \subseteq C$ is the set of all states of C that are included in any path in $TC\{i, j\}$ for any i. When the set C does not include s_{Term}, or when $j = Term$, each path in the trajectory $TC\{i, j\}$ occurs entirely within a single test case. In this instance, the trajectory with respect to the set C is called a *test trajectory*, denoted $TT\{i, j\}$. For example, in the case of Figure 2, when $C = \{s_2, s_8, s_9, s_{12}\}$,

$TT\{2, 2\} = \{\}$ (i.e., the empty trajectory),

$TT\{2, 8\} = \{s_2 s_8\}$,

$TT\{2, 12\} = \{s_2 s_8 s_9 s_{12}, s_2 s_8 s_9 s_2 s_8 s_9 s_{12}, \ldots\}$,

etc.

The *entropy of the trajectory with respect to a set* C, denoted $H(TC\{i, j\})$, can be interpreted as the minimum average number of "yes or no" questions to determine which path was chosen from the trajectory $TC\{i, j\}$. Thus, for all $t_{i,j} \in TC\{i, j\}$, $H(TC\{i, j\}) = -\sum_{t_{i,j}} prob(t_{i,j}) \log(prob(t_{i,j}))$. Note that this trajectory entropy is undefined when the trajectory is an empty set. Similarly, $H(TT\{i, j\})$ is the *entropy of the test trajectory* $TT\{i, j\}$.

Because the number of paths in $TC\{i, j\}$ might be infinite, the above formula cannot be applied directly. As noted in [8], an equivalent definition for trajectory entropy

is the sum of the uncertainty of the first step in the path plus the conditional uncertainty of the rest of the path, given the first step.

Theorem 4.1 The uncertainty in choosing a path from s_{Inv} to s_{Term} can be calculated as follows:

$$H(TT\{Inv, Term\}) = \frac{H\{M\}}{\pi_{Inv}}$$

where $H\{M\}$ is the source entropy of P, and π_{Inv} is the entry in the stationary distribution vector for s_{Inv}.

Proof: For Markov chain usage models, $p_{Term, Inv} = 1$. Thus, since there is only one path from s_{Term} to s_{Inv}, $H(TT\{Inv, Term\}) = H(TT\{Inv, Inv\})$. Theorem 1 of [8] proves $H(TT\{Inv, Inv\}) = \frac{H\{M\}}{\pi_{Inv}}$. □

Theorem 4.2 Let P, $TC\{i, j\}$ and $C\{j\}$ be as defined above.

Construct P' from P as follows:

(1) delete row k and column k, for every k such that $s_k \notin C\{j\}$

(2) change row j so that $p_{j, i} = 1/m$ where m = size of $C\{j\}$

(3) normalize the resulting matrix so that the rows sum to 1

Construct $\tilde{P}$ from P' by deleting row j and column j.

Then for each $j \in \{1, \ldots, n\}$,

(1) P' is a row-stochastic, irreducible matrix,

(2) $(I - \tilde{P})^{-1}$ exists, and

(3) the vector of trajectory entropies is given by

$[H(TC\{i,j\})]_{i\neq j} = (I-\tilde{P})^{-1}[h'(s_i)]_{i\neq j}$ for $TC\{i,j\} \neq \varnothing$, where $i \in \{1, \ldots, n\}$, $(i \neq j)$ and $h(s_i')$ is the entropy of row i of the matrix P'.

Proof: (1) Since every row sums to 1, P' is a row-stochastic matrix of size m (where m = size of C{j}). P' only includes those states s_k' such that $s_k' \in C\{j\}$. Thus, for every state s_k' in the Markov chain represented by P' there exists a path from state s_k' to state s_j'. Since the construction of P' included a path from s_j' to s_k', every state in the Markov chain is reachable from every state. Therefore P' is irreducible.

(2) The ordering of the states in a Markov chain usage model is mathematically irrelevant. Number the states from set C{j}, excluding state s_j, as states $s_1, s_2, \ldots, s_{m-1}$, and add a state s_m such that $p_{m,1} = 1$ and all other $p_{m,i} = 0$. The result is a row-stochastic irreducible square matrix of size *m*, where *m* = size of the set C{j}. Thus $\tilde{P}$ is a principal submatrix of a row-stochastic, irreducible matrix. Therefore, $(I-\tilde{P})^{-1}$ exists and is non-negative (Theorem 5.1 of [17]).

(3) For random variables X and Y,

$H(X, Y) = H(Y|X) + H(X)$ and $H(Y|X) = \sum_{k=1}^{n} \text{Prob}(X = x_k)\, H(Y|X = x_k)$ [4].

If the X is the selection of the first step and Y is the selection of the rest of the path after the first step, $H(TC\{i,j\})$ must be equal to the uncertainty of the first step plus the conditional uncertainty of the rest of the path, given the first step. The first step can only consist of states in C{j}. If $k = j$, the entire path is defined after the first step; thus, because there is no uncertainty with respect to the rest of the path, $H(Y|X = x_k) = 0$.

Therefore, for $i \in \{1, \ldots, n\}$, $i \neq j$, and $TC\{i,j\} \neq \varnothing$,

$[H(TC\{i,j\})] = \tilde{P}[H(TC\{i,j\})] + [h(s_i')]$

giving $[H(TC\{i,j\})] = (I - \tilde{P})^{-1}[h(s_i')]$ □

4.2.2 Calculating Specification Complexity

Theorem 4.2 provides a closed form solution for calculating $H(TC\{i,j\})$ which requires m-1 matrix inversions, where m is the number of states in the set $C\{j\}$. When $C = S$, a closed form solution is provided by [8] that is based on the classical "fundamental matrix" of Kemeny and Snell [16].

For the case $C = S$, both the exposition and the computational properties of formulas based on the group inverse of Meyer [17] are preferable. The work of [8] is modified to use the group inverse as follows:

Theorem 4.3 Consider the transition matrix P for an n-state irreducible Markov chain. The matrix of trajectory entropies as defined in [8] can be calculated by:

$$H(T\{i,j\}) = \left(\sum_{k=1}^{n} \left(a^{\#}_{i,k} - a^{\#}_{j,k} \right) H(s_k) \right) - \left(a^{\#}_{i,j} - a^{\#}_{j,j} \right) H(T\{j,j\})$$

where $a^{\#}_{i,j}$ is the i,j-th element of $A^{\#}$, and $H(s_k)$ is the state entropy for state s_k.

Proof: [8] proves that $H(T\{i,j\}) = K - \tilde{K} + H_{\Delta}$, where

$$K = (I - P + W)^{-1}\left(H^{*} - H_{\Delta} \right)$$

$\tilde{K}_{i,j} = K_{jj}$ for all i,j

$W_{i,j} = \pi_j$ for all i,j

π_j is the j-th element of the stationary distribution of P

$H^*_{i,j} = H(s_i)$ for all i,j

$(H_\Delta)_{i,j} = (H\{M\})/\pi_i$ for $i = j$

$(H_\Delta)_{i,j} = 0$ for $i \neq j$

[16] proves that $(I - P + W)^{-1} = A^{\#} + W$. We revise the proof in [8] accordingly:

Let $K = \left(A^{\#} + W\right)\left(H^* - H_\Delta\right)$

Then $K_{i,j} = \sum_k \left(a^{\#}_{i,k} + \pi_k\right)(H(s_k) - (H_\Delta)_{k,j})$

and $K_{i,j} = \left(\sum_k \left(a^{\#}_{i,k} + \pi_k\right)H(s_k)\right) - \left(a^{\#}_{i,j} + \pi_j\right)H(T\{i,j\})$

But $H\{M\} = \sum_k \pi_k H(s_k)$ and $H(T\{i,j\}) = H\{M\}/\pi_j$

Therefore, $K_{i,j} = \left(\sum_k a^{\#}_{i,k} H(s_k)\right) + H\{M\} - a^{\#}_{i,j}(H\{M\}/\pi_j) - \pi_j H\{M\}/\pi_j$

and $K_{i,j} = \left(\sum_k a^{\#}_{i,k} H(s_k)\right) - a^{\#}_{i,j} H(T\{i,j\})$

Thus, for $i \neq j$,

$$H(T\{i,j\}) = \left(\sum_k a^{\#}_{i,k} H(s_k)\right) - a^{\#}_{i,j} H(T\{i,j\}) - \left(\left(\sum_k a^{\#}_{j,k} H(s_k)\right) - a^{\#}_{j,j} H(T\{i,j\})\right)$$

$$\text{Thus, } H(T\{i,j\}) = \left(\sum_k \left(a^{\#}_{i,k} - a^{\#}_{j,k}\right) H(s_k)\right) - \left(a^{\#}_{i,j} - a^{\#}_{j,j}\right) H(T\{i,j\}) \quad \square$$

4.2.3 Interpretation and Illustration

Source entropy has been used to compare two Markov chain usage models by examining the number of typical long-run sequences for the models [27, 29, 30]. In addressing questions concerning model coverage during statistical testing, it is appropriate to compare two usage models with respect to the number of canonical paths between the Invocation and Termination states. Since $H(TT\{i,j\})$ is the minimum average number of "yes or no" questions to determine which path was chosen from the test trajectory $TT\{i,j\}$, the number of possible paths in the test trajectory is on the order of $2^{H(TT\{i,j\})}$. An increase in $H(TT\{Inv, Term\})$ results in an exponential increase in the number of canonical test paths.

Test trajectory entropy is a measure of uncertainty related to the number of canonical paths in the model which is in turn a measure of the complexity of the specification. The entropy of the test trajectory with respect to any subset of states provides a measure of the uncertainty in selecting a single test case that includes only the corresponding subset of the specification. Because this measure can be generated with respect to an arbitrary subset of states in the model, it can be used to identify the more complex sections of the specification.

For both versions of the 15-state usage model of Figure 2, Table 5 gives the matrix of trajectory entropies as defined by [8]. For the model with uniform transition probabilities, $H(T\{Inv, Term\}) = 19.1$, but for the model with non-uniform transition probabilities $H(T\{Inv, Term\}) = 11.1$. The second version has fewer canonical test paths than the first. The entries in Table 5 provide a measure of the uncertainty in selection of a path between two states where that path occurs within a long-run chain of states generated by the usage model.

Table 6 gives the matrix of test trajectory entropies where $C = S$. Blank cells in the matrices indicate that there are no paths in the corresponding trajectory. Our definition of trajectory with respect to a set C and definition of test trajectory do not include paths that begin and end in the same state. Each cell on the diagonal of each matrix is undefined. The entries in this table provide a measure of the uncertainty in selection of a path between two states where that path occurs within a single test case. Note that, except for the last two rows, the right hand column of each matrix corresponds to those of Table 5. The value for each element of a given row of Table 6 is less than or equal to the corresponding value of the right-hand element of the corresponding row.

Path coverage analysis can provide insight into the testing process that is not available from analysis of state and arc coverage. For example, the matrix of expected number of test cases to achieve arc coverage shown in Table 4 indicates that in Version 2, some of the arcs representing transitions from any of $\{s_2,...,s_7\}$ to any of $\{s_2,...,s_7\}$ will require more testing (on average) before the arc is covered, and some will be covered (on average)

Table 5: Trajectory Entropy

Version 1: Uniform Probabilities

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
s1	19.1	0.	23.9	23.9	11.9	11.9	23.9	11.6	12.6	44.5	43.0	38.3	20.3	19.1	[illegible]
s2	19.1	4.8	23.9	23.9	11.9	11.9	23.9	11.6	12.6	44.5	43.0	38.3	20.3	19.1	19.1
s3	19.1	0.	23.9	23.9	11.9	11.9	23.9	11.6	12.6	44.5	43.0	38.3	20.3	19.1	19.1
s4	19.1	0.	23.9	23.9	0.	0.	23.9	11.6	12.6	44.5	43.0	38.3	20.3	19.1	19.1
s5	19.1	0.	23.9	23.9	11.9	0.	23.9	11.6	12.6	44.5	43.0	38.3	20.3	19.1	19.1
s6	19.1	0.	23.9	23.9	11.9	11.9	23.9	11.6	12.6	44.5	43.0	38.3	20.3	19.1	19.1
s7	19.1	0.	23.9	23.9	0.	0.	23.9	11.6	12.6	44.5	43.0	38.3	20.3	19.1	19.1
s8	26.6	12.3	36.2	36.2	24.2	24.2	36.2	14.7	1.0	32.8	31.3	26.7	8.6	26.6	26.6
s9	25.6	11.3	35.2	35.2	23.2	23.2	35.2	13.7	6.4	31.8	30.3	25.7	15.3	25.6	25.6
s10	25.6	11.3	35.2	35.2	23.2	23.2	35.2	13.7	0.	31.8	30.3	25.7	15.3	25.6	25.6
s11	27.1	12.8	36.7	36.7	24.7	24.7	36.7	7.9	1.5	33.3	31.8	27.2	13.0	27.1	27.1
s12	19.0	9.4	33.3	33.3	21.4	21.4	33.3	10.0	6.1	38.0	36.5	31.8	16.3	19.0	19.0
s13	25.6	11.3	35.2	35.2	23.2	23.2	35.2	13.7	0.	31.8	30.3	25.8	15.3	25.6	25.6
s14	0.	0.	23.9	23.9	11.9	11.9	23.9	11.6	12.6	44.5	43.0	38.3	20.3	19.1	19.1
s15	0.	0.	23.9	23.9	11.9	11.9	23.9	11.6	12.6	44.5	43.0	38.3	20.2	19.1	19.1

Version 2: Non-Uniform Probabilities for States s2 and s11

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_1	11.1	0.	15.9	15.9	8.0	8.0	15.9	24.4	25.4	73.7	72.7	63.3	37.9	11.1	[illegible]
s_2	11.1	3.34	15.9	15.9	8.0	8.0	15.9	24.4	25.4	73.7	72.7	63.3	37.9	11.1	11.1
s_3	11.1	0.	15.9	15.9	8.0	8.0	15.9	24.4	25.4	73.7	72.7	63.3	37.9	11.1	11.1
s_4	11.1	0.	15.9	15.9	0.	0.	15.9	24.4	25.4	73.7	72.7	63.3	37.9	11.1	11.1
s_5	11.1	0.	15.9	15.9	8.0	0.	15.9	24.4	25.4	73.7	72.7	63.3	37.9	11.1	11.1
s_6	11.1	0.	15.9	15.9	8.0	8.0	15.9	24.4	25.4	73.7	72.7	63.3	37.9	11.1	11.1
s_7	11.1	0.	15.9	15.9	.0	0.	15.9	24.4	25.4	73.7	72.7	63.3	37.9	11.1	11.1
s_8	20.2	11.8	27.7	27.7	19.8	19.8	27.7	25.9	1.0	49.3	48.4	38.9	13.5	20.2	20.2
s_9	19.2	10.8	26.7	26.7	18.8	18.8	26.7	24.9	9.7	48.3	47.4	37.9	25.0	19.2	19.2
s_{10}	19.2	10.8	26.7	26.7	18.8	18.8	26.7	24.9	0.	48.3	47.4	37.9	25.0	19.2	19.2
s_{11}	20.1	11.8	27.6	27.6	19.7	19.7	27.6	20.6	0.9	49.2	48.3	38.8	23.4	20.1	20.1
s_{12}	14.7	9.1	25.0	25.0	17.1	17.1	25.0	18.0	10.4	58.6	57.7	48.3	27.0	14.7	14.7
s_{13}	19.2	10.8	26.7	26.7	18.8	18.8	26.7	24.9	0.	48.3	47.4	37.9	25.0	19.2	19.2
s_{14}	0.	0.	15.9	15.9	7.9	7.9	15.9	24.4	25.4	72.7	72.7	63.3	37.9	11.1	11.1
s_{15}	0.	0.	15.9	15.9	7.9	7.9	15.9	24.4	25.4	72.7	72.7	63.3	37.9	11.1	11.1

Table 6: Test Trajectory Entropy, Entire Model

Version 1: Uniform Probabilities

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_1		0.	9.7	9.7	6.7	6.7	9.7	5.0	5.5	10.9	10.5	11.7	7.5	19.1	19.1
s_2			9.7	9.7	6.7	6.7	9.7	5.0	5.5	10.9	10.5	11.7	7.5	19.1	19.1
s_3		0.		9.7	6.7	6.7	9.7	5.0	5.5	10.9	10.5	11.7	7.5	19.1	19.1
s_4		0.	9.7		0.	0.	9.7	5.0	5.5	10.9	10.5	11.7	7.5	19.1	19.1
s_5		0.	9.7	9.7		0.	9.7	5.0	5.5	10.9	10.5	11.7	7.5	19.1	19.1
s_6		0.	9.7	9.7	6.7		9.7	5.0	5.5	10.9	10.5	11.7	7.5	19.1	19.1
s_7		0.	9.7	9.7	0.	0.		5.0	5.5	10.9	10.5	11.7	7.5	19.1	19.1
s_8		11.8	19.1	19.1	17.0	17.0	19.1		1.0	11.9	11.1	13.4	5.0	26.6	26.6
s_9		10.8	18.1	18.1	15.9	15.9	18.1	8.6		10.9	10.1	12.4	8.0	25.6	25.6
s_{10}		10.8	18.1	18.1	15.9	15.9	18.1	8.6	0.		10.1	12.4	8.0	25.6	25.6
s_{11}		12.8	19.6	19.6	17.4	17.4	19.6	5.3	1.5	12.4		13.9	7.5	27.1	27.1
s_{12}		8.6	13.4	13.4	11.9	11.9	13.4	3.9	1.3	8.6	8.0		5.3	19.0	19.0
s_{13}		10.8	18.1	18.1	15.9	15.9	18.1	8.6	0.	10.9	10.1	12.4		25.6	25.6
s_{14}															0.
s_{15}															

Version 2: Non-Uniform Probabilities for States s2 and s11

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
s1		0.	5.9	5.9	4.1	4.1	5.9	5.9	5.5	7.8	7.7	8.1	6.5	11.1	11.1
s2			5.9	5.9	4.1	4.1	5.9	5.9	5.5	7.8	7.7	8.1	6.5	11.1	11.1
s3		0.		5.9	4.1	4.1	5.9	5.9	5.5	7.8	7.7	8.1	6.5	11.1	11.1
s4		0.	5.9		0.	0.	5.9	5.9	5.5	7.8	7.7	8.1	6.5	11.1	11.1
s5		0.	5.9	5.9		0.	5.9	5.9	5.5	7.8	7.7	8.1	6.5	11.1	11.1
s6		0.	5.9	5.9	4.1		5.9	5.9	5.5	7.8	7.7	8.1	6.5	11.1	11.1
s7		0.	5.9	5.9	0.	0.		5.9	5.5	7.8	7.7	8.1	6.5	11.1	11.1
s8		11.4	15.8	15.8	14.5	14.5	15.8		1.0	10.3	9.9	11.3	4.9	20.2	20.2
s9		10.4	14.8	14.8	13.5	13.5	14.8	14.8		9.3	8.9	10.3	7.8	19.2	19.2
s10		10.4	14.8	14.8	13.5	13.5	14.8	14.8	0.		8.9	10.3	7.8	19.2	19.2
s11		11.3	15.7	15.7	14.4	14.4	15.7	15.7	0.9	10.2		11.2	8.0	20.1	20.1
s12		8.3	11.2	11.2	10.3	10.3	11.2	11.2	1.3	7.5	7.2		5.2	14.7	14.7
s13		10.4	14.8	14.8	13.5	13.5	14.8	14.8	0.	9.3	8.9	10.3		19.2	19.2
s14															0.
s15															

earlier than in Version 1. However, further examination of the matrix of test trajectory entropies in Table 6 shows that this analysis of arc coverage does not provide sufficient information concerning the amount of testing required to cover canonical test paths in sub-sections of the model. The matrix of test trajectory entropies shows that while this specific change to non-uniform transition probabilities for states s_2 and s_{11} did reduce $H(TT\{i,j\})$ for all i,j where $j \neq 8$, it increased $H(TT\{i,j\})$ for $j = 8$.

The next two tables provide more information that can be used to analyze path complexities for the two major subsections of the usage model. Table 7 provides the matrix of test trajectory entropies for the upper section of the usage model, states $s_2, \ldots, s_7$, and Table 8 provides the matrix of test trajectory entropies for the lower section of the usage model, states $s_2, s_8, s_9, \ldots, s_{14}$. A comparison of these two tables shows quantitatively that, in general, the lower section of the usage model is more complex than the upper section.

4.3 Optimization

The problem of selecting test cases so that testing traverses each arc at least once with the fewest total number of transitions is an instance of the Chinese postman problem for directed graphs [9]. This is of interest as a means of covering the structure of the usage model; however, it does not address statistically typical paths of use. A previous chapter demonstrated the general applicability of convex programming to the representation and optimization of usage models. This section offers a collection of useful objective functions and constraints based on the above analysis of coverage and complexity.

Table 7: Test Trajectory Entropy, Upper Section

Version 1: Uniform Probabilities

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1		0.	2.64	2.64	1.98	1.98	2.64
s_2			2.64	2.64	1.98	1.98	2.64
s_3		0.		2.64	1.98	1.98	2.64
s_4		0.	2.64		0.	0.	2.64
s_5		0.	2.64	2.64		0.	2.64
s_6		0.	2.64	2.64	1.98		2.64
s_7		0.	2.64	2.64	0.	0.	

Version 2: Non-Uniform Probabilities for States s2 and s11

	s1	s2	s3	s4	s5	s6	s7
s_1		0.	2.74	2.74	2.01	2.01	2.74
s_2			2.74	2.74	2.01	2.01	2.74
s_3		0.		2.74	2.01	2.01	2.74
s_4		0.	2.74		0.	0.	2.74
s_5		0.	2.74	2.74		0	2.74
s_6		0.	2.46	2.74	2.01		2.74
s_7		0.	2.74	2.74	0.	0.	

Table 8: Test Trajectory Entropy, Lower Section

Version 1: Uniform Probabilities

	s_2	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}
s_2		0.	0.20	1.57	1.43	1.68	0.75	4.06
s_8	11.83		1.00	7.86	7.16	8.39	3.76	15.32
s_9	10.83	6.28		6.86	6.16	7.39	5.53	14.32
s_{10}	10.83	6.28	0.		6.16	7.39	5.53	14.32
s_{11}	12.33	4.14	1.50	8.36		8.89	5.64	15.82
s_{12}	8.56	3.09	1.33	5.91	5.44		4.10	11.46
s_{13}	10.83	6.28	0.	6.86	6.16	7.39		14.32
s_{14}								

Version 2: Non-Uniform Probabilities for States s2 and s11

	s_2	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}
s_2		0.	0.09	0.67	0.63	0.70	0.34	2.05
s_8	11.40		1.00	7.22	6.83	7.62	3.73	13.37
s_9	10.40	7.14		6.22	5.82	6.62	5.47	12.37
s_{10}	10.40	7.14	0.		5.82	6.62	5.47	12.37
s_{11}	11.32	6.44	0.92	7.15		7.54	5.85	13.29
s_{12}	8.27	3.38	1.33	5.49	5.21		4.07	10.17
s_{13}	10.40	7.14	0.	6.23	5.82	6.62		12.37
s_{14}								

4.3.1 Theorems Concerning Constraints

Theorem 4.4 Consider unknown transition matrix P, $0 \le \alpha_{i,j} \le p_{i,j} \le \beta_{i,j} \le 1$, for an n-state irreducible Markov chain. Then:

(1) the stationary distribution vector π of P is bounded by the linear constraints

$$\sum_{i=1}^{n} \pi_i \alpha_{i,j} - \pi_j \le 0 \text{ for each } j$$

$$\sum_{i=1}^{n} \pi_i \beta_{i,j} - \pi_j \ge 0 \text{ for each } j$$

$$\pi_j > 0 \text{ for each } j$$

$$\sum_{j} \pi_j = 1$$

and (2) for any π satisfying these constraints, there exists a row-stochastic, irreducible matrix P such that π is the stationary distribution for P.

Proof: This theorem is proved in [24]. ❐

Theorem 4.5 Consider unknown transition matrix P, $0 \le \alpha_{i,j} \le p_{i,j} \le \beta_{i,j} \le 1$, for an n-state irreducible Markov chain. Then:

(1) for every i,j one can tighten the bounds for $p_{i,j}$ so that:

$$\alpha_{i,j} \ge 1 - \sum_{k \ne j} \beta_{i,k}$$

$$\beta_{i,j} \le 1 - \sum_{k \ne j} \alpha_{i,k}$$

and (2) $y_{Inv,j}$, the probability that state s_j occurs in a test case, is bounded by the linear constraints:

$$0 \le y_{i,j} \le 1 \text{ for } i \in \{1, \ldots, n\}$$

$$y_{j,j} = 0$$

$$y_{Term,j} = 0$$

$$-y_{i,j} + \alpha_{i,j} + \sum_{k \notin \{j, Term\}} \alpha_{i,k}\, y_{k,j} \le 0 \text{ for } i, k \in \{1, \ldots, n\}$$

$$-y_{i,j} + \beta_{i,j} + \sum_{k \notin \{j, Term\}} \beta_{i,k}\, y_{k,j} \ge 0 \text{ for } i, k \in \{1, \ldots, n\}$$

Proof: For any fixed initial n-state irreducible row-stochastic matrix P^0 such that $\alpha_{i,j} \le p^0_{i,j} \le \beta_{i,j}$, there must exist a matrix δP of size n such that:

$$P^0 + \delta P = P$$

and $\sum_{j=1}^{n} \delta p_{i,j} = 0$ for every row i.

(1) $p^0_{i,j} \ge 1 - \sum_{k \ne j} \beta_{i,k}$

and $p^0_{i,j} \le 1 - \sum_{k \ne j} \alpha_{i,k}$.

Thus, if $\alpha_{i,j} \ge 1 - \sum_{k \ne j} \beta_{i,k}$ for each i,j

and $\beta_{i,j} \le 1 - \sum_{k \ne j} \alpha_{i,k}$ for each i,j,

then the bounds on P are tight.

(2) P is irreducible and row stochastic. Therefore $y_{Inv,j}$, the probability that state s_j occurs in a test case, exists and can be calculated by:

$$y_{i,j} = q_{i,j} + \sum_{k \notin \{j, Term\}} q_{i,k} y_{k,j} \qquad \text{(Eq 4.1)}$$

$$y_{j,j} = 1$$

$$y_{Term,j} = 0$$

$$0 \le y_{i,j} \le 1$$

where Q is the transition matrix for an absorbing Markov chain defined by P with absorbing states Term and j [16].

Proof that the constraints bound $y_{Inv,j}$ follows directly the techniques of the proof given in [24]:

Q^0 is the transition matrix for an absorbing Markov chain defined by P^0 with absorbing states Term and j.

δQ is defined by $Q^0 + \delta Q = Q$

By Eq 4.1, $-y_{i,j} + \left(q^0_{i,j} + \delta q_{i,j}\right) + \sum_{k \notin \{j, Term\}} \left(q^0_{i,k} + \delta q_{i,k}\right) y_{k,j} = 0$ (Eq 4.2)

Therefore, $-y_{i,j} + q^0_{i,j} + \sum_{k \notin \{j, Term\}} q^0_{i,k} y_{k,j} + \delta q_{i,j} + \sum_{k \notin \{j, Term\}} \delta q_{i,k} y_k = 0$

By definition of Q^0,

$$\alpha_{i,j} - q^0_{i,j} \le \delta q_{i,j} \le \beta_{i,j} - q^0_{i,j}$$

and $$-\left(q^0_{i,j} - \alpha_{i,j}\right) \le \delta q_{i,j} \le \beta_{i,j} - q^0_{i,j}$$

Thus, $\left(\delta q_{i,j} + \sum_{k \notin \{j, Term\}} \delta q_{i,k} y_{k,j}\right) \geq -\left(q^0_{i,j} - \alpha_{i,j} + \sum_{k \notin \{j, Term\}} \left(q^0_{i,k} - \alpha_{i,k}\right) y_{k,j}\right)$

and $\left(\delta q_{i,j} + \sum_{k \notin \{j, Term\}} \delta q_{i,k} y_k\right) \leq \left(\beta_{i,j} - q^0_{i,j} + \sum_{k \notin \{j, Term\}} \left(\beta_{i,k} - q^0_{i,k}\right) y_{k,j}\right)$

By Eq 4.2,

$$-y_{i,j} + q^0_{i,j} + \sum_{k \notin \{j, Term\}} q^0_{i,k} y_{k,j} - \left(q^0_{i,j} - \alpha_{i,j} + \sum_{k \notin \{j, Term\}} \left(q^0_{i,k} - \alpha_{i,k}\right) y_{k,j}\right) \leq 0$$

and

$$-y_{i,j} + q^0_{i,j} + \sum_{k \notin \{j, Term\}} q^0_{i,k} y_{k,j} + \left(\beta_{i,j} - q^0_{i,j} + \sum_{k \notin \{j, Term\}} \left(\beta_{i,k} - q^0_{i,k}\right) y_{k,j}\right) \geq 0$$

Thus, $-y_{i,j} + \alpha_{i,j} + \sum_{k \notin \{j, Term\}} \alpha_{i,k} y_{k,j} \leq 0$

and $-y_{i,j} + \beta_{i,j} + \sum_{k \notin \{j, Term\}} \beta_{i,k} y_{k,j} \geq 0$ □

4.3.2 Formulation of Convex Programming Problems

The general optimization problems of interest (not directly convex) with respect to usage model coverage are:

- Find the irreducible, row-stochastic matrix P that maximizes (or minimizes) both $(1/\pi_{Inv})$ and $(1/y_{Inv,j})$, i.e., minimizes (or maximizes) the expected test case length and the number of test cases required before state s_j occurs at least once.
- Find the irreducible, row-stochastic matrix P that maximizes (or minimizes)

$(1/\pi_{Inv})$, $(1/y_{Inv,j})$, and $(1/p_{j,k})$, i.e., minimizes (or maximizes) the expected test case length and the number of test cases before arc (j,k) occurs at least once.

- Find the irreducible, row-stochastic matrix P that maximizes (or minimizes) $(H\{M\}/\pi_{Inv})$, i.e., maximizes (or minimizes) the number of canonical paths from s_{Inv} to s_{Term}.

A specific vector π may be the stationary distribution for many row-stochastic, irreducible matrices P. Theorem 4.4 and Theorem 4.5 demonstrate that linear programming can be used to bound possible values of π_{Inv}, $y_{Inv,j}$ (any j), and $p_{i,j}$ (any i,j) as required in the first two of the above problems.

The third problem becomes a convex programming problem if the stationary distribution is fixed. When the vector π is fixed, π_i/π_{Inv} is a constant. Thus,

$$\frac{\partial}{\partial p_{i,j}} H(TT\{Inv, Term\}) = -\left(\frac{\pi_i}{\pi_{Inv}}(\log(p_{i,j}) + 1)\right)$$

$$\frac{\partial^2}{\partial p_{i,j}^2} H(TT\{Inv, Term\}) = -\left(\frac{\pi_i}{\pi_{Inv}}\right)\left(\frac{1}{p_{i,j}}\right) < 0 \text{ and}$$

$$\frac{\partial^2}{\partial p_{a,b}\partial p_{i,j}} H(TT\{Inv, Term\}) = 0 \text{ if } a, b \neq i, j.$$

This demonstrates that the Hessian of $H(TT\{Inv, Term\})$ is negative definite; meaning that the function is strictly concave. Therefore, convex programming techniques can be used to find, for a fixed stationary distribution π, the unique transition matrix P which yields the maximum number of canonical test paths.

The following are examples of the formulation of convex programming problems that are appropriate for use when the amount of planned testing is sufficient to yield a set of test cases that includes the canonical paths of the usage model. In the following, F is a convex objective function and constraint functions in the set G are convex.

- To generate a vector π such that the expected test case length of P is minimum:

  Maximize π_{Inv} subject to the constraints given in Theorem 4.4

- To maximize the expected probability of occurrence of each state during testing where each state is equally important, use the following procedure:

  Let $\pi = \{\pi_1, \ldots, \pi_n\}$

  Let G be a set of constraints including the constraints given in Theorem 4.4 (and possibly other constraints.)

  For m = 1 to n-1:

  { Let μ be the minimum element of π and let ι be its index.

  Maximize μ subject to G to yield constant c.

  $G := G \cup \{\pi_\iota = c\}$

  $\pi := \pi - \{\pi_\iota\}$

  }

- To generate a transition matrix P to correspond to a specified stationary distribution vector π (for example, this vector might have been selected by one of the pre-

vious two examples):

Maximize (or minimize) F subject to:

$$\sum_{j} p_{i,j} = 1 \text{ for every row } i$$

$$\pi_j = \sum_{i} \pi_i \, p_{i,j}$$

- To generate a transition matrix P that optimizes F subject to the constraints that the expected test case length is no more than some constant c_1 and a particular state s_i is expected to occur in at least some constant c_2 percent of the test cases:

  Maximize (or minimize) F subject to:

  the set of constraints given in Theorem 4.4

  $$\pi_{Inv} \geq \frac{1}{c_1}$$

  $$\pi_i \geq \frac{c_2}{100} \frac{1}{c_1}$$

- For specified stationary distribution, π, to generate a transition matrix P that will maximize the expected probability of occurrence of each exit arc from a specified set of m states, where each arc is equally important (i.e., we want to maximize $\pi_j \, p_{j,k}$, for every $k \in \{1, \ldots, n\}$ and each state s_j of interest), use the following procedure:

  Let $\Omega = \{\pi_j \, p_{j,k}\}_{j \in \{1, \ldots, n\}, k \in \{1, \ldots, n\}}$

  Let constraint set G include the union of the set of constraints given in

Theorem 4.5 and the set of constraints specifying the values for each of the π_j

For m = 1 to n-1:

{ Let μ be the minimum element of Ω with identifying subscripts ι and κ

Maximize $\mu = \pi_\iota p_{\iota,\kappa}$, subject to G to yield constant c.

$G := G \cup \{\pi_\iota p_{\iota,\kappa} = c\}$

$\Omega := \Omega - \{\pi_\iota p_{\iota,\kappa}\}$

}

- For specified stationary distribution π, to generate a transition matrix P that will maximize the number of canonical paths from s_{Inv} to s_{Term}:

  Maximize $(H\{M\})/\pi_{Inv}$ subject to any known convex constraints.

- For specified stationary distribution π, the problem to generate a transition matrix P to minimize the number of canonical paths between states s_i and s_j does not yield a unique solution. The objective function is concave, meaning that there may be multiple solutions for P which yield the minimum number of canonical test paths. The use of convex programming techniques to minimize this function can, in general, be used to yield one of these local minima that meets all other specified objectives and all specified constraints.

Chapter 5

Application of the Maximum Entropy Principle

Convex programming may yield optimal assignments or ranges of optimal assignments for each transition probability in the usage model. When the testing objectives and constraints are satisfied by ranges of optimal assignments for certain transition probabilities, a method is required to assign explicit values to each of these transition probabilities. The maximum entropy estimate is the "least biased estimate possible on the given information; i.e., it is maximally noncommittal with regard to missing information" [13]. This chapter presents the application of the maximum entropy principle to the probability distributions associated with usage models, in order to generate the transition probabilities that satisfy all constraints and are otherwise least biased. A recommended approach is given for the use of maximum entropy in the incremental usage model optimization process described in Figure 1.

5.1 Weighted Mean State Entropy and Source Entropy

Generalizing on the objective function used in the illustration in Chapter 3, we define weighted mean state entropy as follows for a transition matrix of n states:

$$\text{WMSE} = \sum_{i=1}^{n} a_i H(s_i) \quad \text{where the } a_i \text{ are positive constant weights.}$$

Source entropy, H{M}, is π-weighted WMSE. H{M} is not concave in general, so we have no guarantee that the application of convex programming to maximize H{M} will yield a P which gives a global maximum for H{M}. Thus, if the application-appropriate values for π are unknown, one might start the usage model optimization process by maximizing uniform-weighted WMSE and then approximate H{M} by iteratively adding constraints and weights according to testing policy and expected use of the software.

Maximizing weighted mean state entropy (WMSE) with uniform weights is appropriate when the transition probabilities are in question for only a single state or when the transition probabilities are in question for only those states that are equally likely to occur in the long run. It is also the right starting point for usage model optimization if one chooses to begin the process with a baseline model with uniform distribution on the exit arcs for each state.

When states have very different long run probabilities it may not be appropriate to select the transition probabilities by maximizing uniform-weighted WMSE. If the expected probability of occurrence of each state is known, one could assign these probabilities to the weights and maximize WMSE. If the expected probability of occurrence of each state is unknown (often the case in practice), constraints (on test case length, on elements of the π vector, etc.) will work toward reducing the amount of choice remaining in the determination of values of the transition probabilities.

When the weights are fixed, WMSE is a strictly concave function of the transition probabilities. Thus maximizing WMSE subject to convex constraints yields a unique transition matrix which gives a global maximum for WMSE.

5.2 Stationary Distribution Entropy

Maximizing the entropy of the stationary distribution, $H(\pi)$, subject to all known constraints, yields the transition matrix P which has the least biased long-run occupancy of each state. For a test objective of ensuring that the long-run occupancy of each state in the test cases is as uniform as possible, it is appropriate to generate the transition probabilities by maximizing $H(\pi)$ subject to all known constraints.

A transition probability matrix with maximum $H(\pi)$ is of interest as a baseline usage model or for comparison to the usage model generated by maximizing WMSE, or any other objective function, because it can indicate which transition probabilities may be responsible for large differences between entries in the π vector.

$H(\pi)$ is a strictly concave function of π. However, there can be infinitely many transition matrices that correspond to a given stationary distribution vector and satisfy the constraints. Thus maximizing $H(\pi)$ yields a unique π which gives a global maximum for $H(\pi)$, but it may not yield a unique P.

5.3 Test Trajectory Entropy

Maximizing the entropy of the test trajectory from Invocation to Termination, abbreviated H(TT), yields the transition matrix P which has the least biased long run average selection of test cases. Because $H(TT) = (H\{M\})/\pi_{Inv}$, maximizing H(TT) may result in a very small value for π_{Inv}, and therefore may result in a very large value for expected test case length. In practice, the test constraints should include an upper bound for expected test case length.

For a fixed stationary distribution, H(TT) is a strictly concave function of P. Thus, convex programming theory is applicable to the problem of finding a global maximum for H(TT) subject to specified values for π. However, π is an eigenvector of P. Thus the constraint set includes:

- P must be of a fixed structure,
- Each row of P must sum to 1,
- P must have a specified eigenvector associated with an eigenvalue equal to 1.

A change to a single $p_{i,j}$ will violate the constraint that row i must sum to 1 and will violate the eigenvector constraint. Thus, the problem is tightly constrained and may not be well scaled. For practical application of convex programming solvers to the maximization of H(TT), additional analysis may be required for selection of an effective optimization algorithm.

Test trajectory entropy is of interest when generating test cases for operational testing of software, typically the last test phase prior to release of the software. The variation in H(TT) for different transition matrices can be examined to analyze the impact of adding constraints and to facilitate the choice between models.

5.4 Illustration

The 15-state usage model with structure as depicted in Figure 2 is used to illustrate the implications of maximizing the entropy of the various probability distributions associated with a usage model. Consider the following constraint set G:

- $g_1(P)$ = structural constraints:
  - The structure must be as specified in Figure 2;
  - For each $p_{i,j}$ not defined by the structure, $0 < \alpha_{i,j} \leq p_{i,j} \leq \beta_{i,j} < 1$, where $\alpha_{i,j} = 0.0001$ and $\beta_{i,j} = 0.9999$;
  - Each row of P must sum to 1.
- $g_2(P) = p_{12,14} - 0.25 \leq 0$
- $g_3(P) = p_{11,8} - p_{12,8} = 0$
- $g_4(\pi) = 0.05 \leq \pi_1 \leq 0.07$
- $g_5(\pi) = \pi_9 - 0.14 = 0$

Note that constraints g_2 and g_3 are functions of the transition probabilities. Constraint g_2 is a single-state constraint, and constraint g_3 establishes a dependency between states s_{11} and s_{12}. The next two constraints are functions of π. Constraint g_4 establishes bounds on the long run probability of the occurrence of the Invocation state and therefore constrains expected test case length. Constraint g_5 fixes the value of a single element of π. (For WMSE, fixing the value of an element of π could have been represented as a change to the weights in the objective function instead of a constraint.)

Table 9 presents the progression of transition probabilities obtained by maximizing uniform-weighted WMSE, subject to the constraint set G. Table 10 presents the progression of the stationary distribution vector π obtained from maximizing uniform-weighted WMSE subject to the constraint set. Table 11 presents the progression of transition probabilities obtained from maximizing $H(\pi)$ subject to the constraint set. Table 12 presents

Table 9: Progression of P for Weighted Mean State Entropy

	WMSE g_1	WMSE & g_2	WMSE & g_3	WMSE & g_4	WMSE & g_5
$p_{2,3}$	0.2000	0.2000	0.2000	0.1987	0.2035
$p_{2,4}$	0.2000	0.2000	0.2000	0.1973	0.2063
$p_{2,7}$	0.2000	0.2000	0.2000	0.1973	0.2065
$p_{2,8}$	0.2000	0.2000	0.2000	0.1949	0.1772
$p_{2,14}$	0.2000	0.2000	0.2000	0.2118	0.2065
$p_{8,9}$	0.5000	0.5000	0.5000	0.5003	0.4995
$p_{8,13}$	0.5000	0.5000	0.5000	0.4997	0.5005
$p_{9,2}$	0.2000	0.2000	0.2001	0.2020	0.2133
$p_{9,10}$	0.2000	0.2000	0.2000	0.1989	0.1951
$p_{9,11}$	0.2000	0.2000	0.2000	0.1985	0.1956
$p_{9,12}$	0.2000	0.2000	0.2000	0.2016	0.2011
$p_{9,13}$	0.2000	0.2000	0.2000	0.1989	0.1950
$p_{11,8}$	0.5000	0.5000	0.4287	0.4284	0.4289
$p_{11,9}$	0.5000	0.5000	0.5713	0.5716	0.5711
$p_{12,8}$	0.3333	0.3750	0.4287	0.4284	0.4289
$p_{12,9}$	0.3334	0.3750	0.3213	0.3216	0.3211
$p_{12,14}$	0.3333	0.2500	0.2500	0.2500	0.2500

Table 10: Progression of π and Entropy Statistics for Weighted Mean State Entropy

	WMSE g_1	WMSE & g_2	WMSE & g_3	WMSE & g_4	WMSE & g_5
π_1	0.0501	0.0473	0.0474	0.0500	0.0500
π_2	0.2005	0.1972	0.1974	0.1995	0.2083
π_3	0.0401	0.0394	0.0395	0.0396	0.0424
π_4	0.0401	0.0394	0.0395	0.0393	0.0430
π_5	0.0802	0.0789	0.0790	0.0787	0.0860
π_6	0.0802	0.0789	0.0790	0.0787	0.0860
π_7	0.0401	0.0394	0.0395	0.0393	0.0430
π_8	0.0652	0.0671	0.0665	0.0653	0.0607
π_9	0.1504	0.1578	0.1579	0.1540	0.1400
π_{10}	0.0301	0.0316	0.0316	0.0306	0.0273
π_{11}	0.0301	0.0316	0.0316	0.0306	0.0274
π_{12}	0.0301	0.0316	0.0316	0.0311	0.0281
π_{13}	0.0626	0.0651	0.0648	0.0633	0.0577
π_{14}	0.0501	0.0473	0.0474	0.0500	0.0500
π_{15}	0.0501	0.0473	0.0474	0.0500	0.0500
WMSE	0.5486	0.5470	0.5453	0.5452	0.5451
H (π)	3.6306	3.6273	3.6271	3.6294	3.6214
H{M}	0.9576	0.9723	0.9716	0.9641	0.9394
H(TT)	19.1050	20.5390	20.5046	19.2829	18.7721

Table 11: Progression of P for Stationary Distribution Entropy

	$H(\pi)$ g_1	$H(\pi)$ & g_2	$H(\pi)$ & g_3	$H(\pi)$ & g_4	$H(\pi)$ & g_5
$p_{2,3}$	0.2076	0.2076	0.2076	0.2075	0.2076
$p_{2,4}$	0.2210	0.2209	0.2209	0.2208	0.2209
$p_{2,7}$	0.2208	0.2208	0.2208	0.2209	0.2209
$p_{2,8}$	0.1245	0.0613	0.0619	0.0656	0.0608
$p_{2,14}$	0.2261	0.2894	0.2888	0.2850	0.2899
$p_{8,9}$	0.0008	0.0001	0.0006	0.0075	0.0002
$p_{8,13}$	0.9992	0.9999	0.9994	0.9925	0.9998
$p_{9,2}$	0.0001	0.0001	0.0001	0.0001	0.0001
$p_{9,10}$	0.3329	0.3331	0.3332	0.3338	0.3333
$p_{9,11}$	0.3331	0.3334	0.3332	0.3331	0.3333
$p_{9,12}$	0.3338	0.3333	0.3334	0.3329	0.3333
$p_{9,13}$	0.0001	0.0001	0.0001	0.0001	0.0001
$p_{11,8}$	0.7289	0.7390	0.7487	0.7470	0.6776
$p_{11,9}$	0.2711	0.2610	0.2513	0.2530	0.3224
$p_{12,8}$	0.5281	0.7599	0.7487	0.7470	0.6776
$p_{12,9}$	0.0001	0.0074	0.0164	0.0032	0.1140
$p_{12,14}$	0.4718	0.2327	0.2349	0.2498	0.2084

Table 12: Progression of π and Entropy Statistics for Stationary Distribution Entropy

	$H(\pi)$ g_1	$H(\pi)$ & g_2	$H(\pi)$ & g_3	$H(\pi)$ & g_4	$H(\pi)$ & g_5
π_1	0.0577	0.0577	0.0577	0.0577	0.0562
π_2	0.1646	0.1646	0.1646	0.1646	0.1602
π_3	0.0342	0.0342	0.0342	0.0342	0.0332
π_4	0.0364	0.0364	0.0364	0.0364	0.0354
π_5	0.0727	0.0727	0.0727	0.0727	0.0707
π_6	0.0727	0.0727	0.0727	0.0727	0.0707
π_7	0.0363	0.0363	0.0363	0.0364	0.0354
π_8	0.0750	0.0750	0.0750	0.0754	0.0730
π_9	0.1300	0.1299	0.1299	0.1298	0.1400
π_{10}	0.0433	0.0433	0.0433	0.0433	0.0467
π_{11}	0.0433	0.0433	0.0433	0.0432	0.0467
π_{12}	0.0434	0.0433	0.0433	0.0432	0.0467
π_{13}	0.0749	0.0750	0.0750	0.0748	0.0730
π_{14}	0.0577	0.0577	0.0577	0.0577	0.0562
π_{15}	0.0577	0.0577	0.0577	0.0577	0.0562
WMSE	0.3821	0.3639	0.3672	0.3677	0.3935
$H(\pi)$	3.7367	3.7367	3.7367	3.7367	3.7352
H{M}	0.6644	0.6406	0.6431	0.6459	0.6730
H(TT)	11.5141	11.0970	11.1402	11.1901	11.9858

the progression of π obtained from maximizing $H(\pi)$ subject to the constraint set. In each table, the column labeled "g_1" is the result of maximizing the function of interest subject to constraint g_1 only. Successive columns to the right incorporate the additional constraint indicated.

Several entropy values are presented in Table 10 and Table 12 to facilitate analysis and comparison of the usage models:

- Weighted mean state entropy with uniform weights, WMSE
- Stationary distribution entropy, $H(\pi)$
- Source entropy, $H\{M\}$
- Test trajectory entropy, H(TT)

As one would expect, the baseline P and associated π (column "g_1" in the tables) for maximum WMSE and for maximum $H(\pi)$ are quite different. A comparison of column "g_1" in each of the tables indicates that the transition probabilities assigned to arcs from states s_8 and s_9 are responsible for some large differences between entries in the π vector.

As constraints are added, the differences between WMSE for the maximum WMSE model and WMSE for the maximum $H(\pi)$ model are reduced. Continuing to fix values of π (or modifying the weights in the WMSE objective function) will continue to reduce the possible values for the transition probabilities. For our example usage model, because of the small number of states and arcs, completely specifying π would determine the value of all the transition probabilities.

It should be noted that convex programming techniques are applicable only with convex constraints. Thus, one can add additional constraints that bound the values of π or constraints that bound the individual transition probabilities. However, because entropy is a concave function, one cannot add a constraint that bounds the values of an entropy function.

5.5 Recommended Process

Of the probability distributions and associated entropies we have discussed, the WMSE is the simplest in concept and is computationally the easiest to use. In addition, as constraints and weights are added to the model, maximum WMSE approximates maximum source entropy. We recommend WMSE as the maximum entropy function of choice for incremental addition of constraints to the usage model to implement the Markov chain usage model optimization process described in Figure 1. The following steps should be followed:

(1) Generate the baseline transition matrix by maximizing uniform-weighted WMSE, subject only to the structural constraints. Calculate WMSE, $H(\pi)$, $H\{M\}$, and H(TT) to facilitate analysis of the constraints and evolving models.

(2) Generate a reference transition matrix by maximizing $H(\pi)$ subject only to the structural constraints. Compare the baseline matrix to the reference matrix to gain insight into the effects of the transition probabilities on the stationary distribution and the entropy statistics.

(3) Add single-state constraints and maximize WMSE. Analyze the impact of adding these constraints.

(4) Add multi-state constraints and maximize WMSE. Analyze the impact of adding these constraints.

(5) Add constraints pertaining to Markov chain statistics (for example, bounds on test case length or specific values for long run probabilities) and maximize WMSE. Analyze the impact of adding these constraints.

(6) Apply objective functions as defined in the test plan to generate usage models appropriate for each phase of testing.

This application of the maximum entropy principle to the iterative process of usage model optimization assures generation of models biased only by the information in the constraints and supports evaluation of the implications of each additional constraint.

Chapter 6

Conclusion

6.1 Summary

Table 13 summarizes the objective functions and constraint functions described in this research. As discussed in [27, 28, 29], the process of building Markov chain usage models admits a great deal of flexibility in abstraction and control of the state space. Models tend to be constructed in layers of detail and complexity from the general to the specific. A state of the chain might be a super-state that is later detailed as a sub-chain. The same super-state might appear several times and be expanded like a macro. Sub-chains of a model can be collapsed to a super-state in order to suppress details. As the model building and analysis progresses, new information is best incorporated in the form of mathematical constraints. The results presented here are supportive of this way of working.

Benefits of using convex programming to optimize transition probabilities include:

- Generating the transition matrix by means of convex programming makes model development and maintenance much more tractable, predictable, and amenable to tool support.
- Optimization allows use of the maximum entropy principle to assign probabilities when multiple matrices satisfy the constraints and objectives. This assures generation of models biased only by the information in the constraints.

Table 13: Summary of Objective Functions and Constraint Functions

Type	Description	Section
Objective	Minimize expected test case length	4.3.2
Objective	Maximize the expected probability of occurrence of each state during testing where each state is equally important	4.3.2
Objective	Maximize expected probability of occurrence of each exit arc from a specified set of states, where each arc is equally important	4.3.2
Objective	For specified stationary distribution, maximize the number of canonical test paths	4.3.2
Objective	For specified stationary distribution, minimize the number of canonical paths beginning at s_i and ending at s_j	4.3.2
Objective	Maximize mean state entropy	5.1
Objective	Maximize stationary distribution entropy	5.2
Objective	Maximize test trajectory entropy	5.3
Constraint	Usage model structure	3.1
Constraint	Basic relationships among arcs	3.1
Constraint	Bounds on the transition probabilities and associated bounds on the stationary distribution vector	4.3.1
Constraint	User-specified stationary distribution vector	4.3.2
Constraint	Expected test case length no more than some constant c_1, and state s_i expected to occur in at least some constant c_2 percent of the test cases	4.3.2

- Post-optimality analysis can assist in understanding the contribution of each set of constraints added to the model.

Specification complexity and coverage measures provide:

- A basis for comparing models and sub-models having the same structure but different parameters,
- Analytical metrics useful to development, test, and maintenance planning,
- Constraints and objective functions useful to the generation of transition matrices,
- A basis to quantify the size of the testing problem.

6.2 Recommendations for future work

This research is immediately applicable to software testing efforts in industry and government. Additional work would be productive in several categories:

- Analyze additional statistics on the usage Markov chain to assess their potential for contribution to the usage model optimization process. For those statistics with potential for contribution, analyze the behavior of the functions to assess their applicability to convex programming.
- Develop and demonstrate additional objective functions and constraints to achieve specific test management goals and to incorporate measures of cost and value associated with testing.
- Describe and demonstrate applicability of weighted states or arcs to usage model optimization.

- Describe and demonstrate procedures for post-optimality analysis of usage model optimization problems.
- Develop and document good engineering practices for usage model development and management.
- Design tool support for these processes.

Bibliography

[1] E. N. Adams, "Optimizing Preventative Service of Software Products", *IBM Journal of Research and Developmen*t, Vol 28, No 1, 1984, pp 2-14.

[2] K. Agrawal and J. A. Whittaker, "Experience in Applying Statistical Testing to a Real-time, Embedded Software System", *Proceedings of the Pacific Northwest Software Quality Conference*, 1994.

[3] T. M. Apostol, *Mathematical Analysis, 2nd edition*, Addison-Wesley, Reading, Massachusetts, 1974.

[4] R. Ash. *Information Theory*, John Wiley and Sons, New York, 1966.

[5] P. H. Ciarlet, *Introduction to Numerical Linear Algebra and Optimization*, Cambridge University Press, 1989.

[6] P. A. Currit, M. Dyer, and H. D. Mills, "Certifying the Reliability of Software, IEEE Transactions on Software Engineering, Vol 12, No 1, January 1986, pp 3-11.

[7] J. W. Duran and S. C. Ntafos, "An Evaluation of Random Testing", *IEEE Transactions on Software Engineering*, Vol 10, No 4, 1984, pp 338-444.

[8] L. Ekroot and T. M. Cover, "The Entropy of Markov Trajectories", *IEEE Transactions on Information Theory*, Vol 39, No 4, July 1993, pp 1418 - 1421.

[9] A. M. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, 1985, p 165.

[10] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, 1981.

[11] G. H. Golub and C. D. Meyer, "Using the QR Factorization and Group Inversion to Compute, Differentiate, and Estimate the Sensitivity of Stationary Probabilities for Markov Chains", *SIAM Journal of Algebraic and Discrete Methods*, Vol 7, 1986, pp 273-281.

[12] D. Hamlet and R. Taylor, "Partition Testing Does Not Inspire Confidence", *IEEE Transactions on Software Engineering*, Vol 16, No 1, 1990, pp 1402-1411.

[13] E. T. Jaynes, "Information Theory and Statistical Mechanics", *Physical Review*, Vol 106, May 1957, pp 620-630.

[14] J. Justesen and T. Hololdt, "Maxentropic Markov Chains", *IEEE Transactions on Information Theory*", Vol 30, No 4, July 1984, pp 665-667.

[15] J. N. Kapur, *Maximum Entropy Models in Science and Engineering*, Wiley, 1989.

[16] J. G. Kemeny and J. L Snell, *Finite Markov Chains*, D. Van Nostrand Company, Inc., 1960.

[17] C. D. Meyer, "The Role of the Group Generalized Inverse in the Theory of Finite Markov Chains", *SIAM Review*, Vol 17, 1975, pp 443-464.

[18] H. D. Mills, "Certifying the Correctness of Software", *Proceedings of the Hawaii International Conference on System Sciences, Vol II, Software Technology*, 1992, pp. 373-281.

[19] J. D. Musa, "Operational Profiles in Software-Reliability Engineering", *IEEE Software*, March 1993, pp 14-32.

[20] W. Oettli, "On the Solution Set of a Linear System with Inaccurate Coefficients", *SIAM Journal of Numerical Analysis, Ser B*, Vol 2, No 1, 1965, pp 115-118.

[21] W. Oettli, W. Prager, and J. H. Wilkinson, "Admissible Solutions of Linear Systems with not sharply defined coefficients", *SIAM Journal of Numerical Analysis*, Ser. B, Vol 2, No 2, pp 291-299.

[22] J. H. Poore, G. H. Walton, J. A. Whittaker, "A Mathematical Programming Approach to the Representation and Optimization of Software Usage Models", In review, June, 1994.

[23] C. E. Shannon, "A Mathematical Theory of Communication", *Bell Systems Technical Journal*, Vol 27, 1948, pp 379-423, 623-656.

[24] J. L. Smith, "Approximate Stationary Probability Vectors of a Finite Markov Chain", *SIAM Journal of Applied Mathematics*, Vol 20, No 4, 1971, pp 612-618.

[25] G. H. Walton and J. H. Poore, "Measuring Complexity and Coverage of Software Specifications", In review, February, 1995.

[26] G. H. Walton, J. H. Poore, and C. J. Trammell, "Statistical Testing of Software Based on a Usage Model", *Software Practice and Experience*, Vol 25, No 1, January, 1995, pp 97-108.

[27] J. A. Whittaker, *Markov Chain Techniques for Software Testing and Reliability Analysis*, Ph.D. Dissertation, Department of Computer Science, University of Tennessee, Knoxville, 1992.

[28] J. A. Whittaker and K. Agrawal, "A Case Study in Software Reliability Measurement", *Proceedings of Quality Week '94*, May, 1994.

[29] J. A. Whittaker and J. H. Poore, "Markov Analysis of Software Specifications", *ACM Transactions on Software Engineering and Methodology*, Vol 2, No 1, pp 93-106, January 1993.

[30] J. A. Whittaker and M. G. Thomason, "A Markov Chain Model for Statistical Software Testing", *IEEE Transactions on Software Engineering*, Vol 30, No 10, pp 812-824, October, 1994.

Appendix

Dr. Mark Pleszkoch of IBM suggested the following approach to the construction of a counter example to the conjecture that source entropy of a Markov chain is concave:

Construct an n-state Markov chain with transition matrix as follows:

- $p_{1,2} = 1$
- $p_{2,2} = a$ for specified $0 < a < 1$
- $p_{2,3} = 1 - a$
- $p_{i,i+1} = 1$ for all $2 < i < n$
- $p_{n,1} = 1$

For $n = 101$, $H\{M\}$ has a global maximum in the interval $0.03 < a < 0.04$, an inflection point in the interval $0.05 < a < 0.06$, and an inflection point in the interval $0.83 < a < 0.84$. Thus $H\{M\}$ is neither convex nor concave.

The graph of $H\{M\}$ versus a for this Markov chain is presented on the following page. The values of $H\{M\}$, the first derivative of $H\{M\}$ with respect to a, and the second derivative with respect to a, for values of a between 0.01 and 0.99 in intervals of 0.01 are presented in the table following the graph.

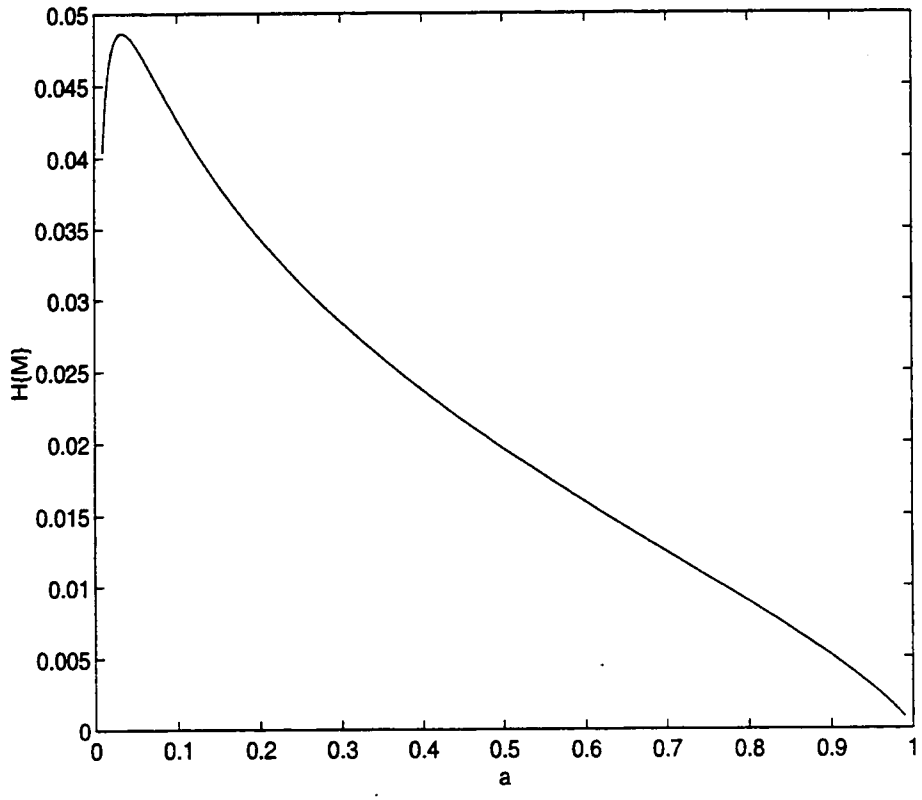

0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
H(M)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
a

Table 14: Source Entropy and Derivatives

a	H{M}	dH{M}/da	$d^2H\{M\}/da^2$
0.0100	0.0404	1.2948	-179.9900
0.0200	0.0471	0.3000	-37.0074
0.0300	0.0486	0.0388	-10.5305
0.0400	0.0485	-0.0522	-3.1213
0.0500	0.0477	-0.0876	-0.5901
0.0600	0.0468	-0.1012	0.3575
0.0700	0.0457	-0.1053	0.7119
0.0800	0.0447	-0.1050	0.8239
0.0900	0.0436	-0.1027	0.8327
0.1000	0.0426	-0.0994	0.7976
0.1100	0.0417	-0.0958	0.7455
0.1200	0.0407	-0.0921	0.6888
0.1300	0.0398	-0.0885	0.6329
0.1400	0.0389	-0.0851	0.5805
0.1500	0.0381	-0.0818	0.5325
0.1600	0.0373	-0.0788	0.4889
0.1700	0.0365	-0.0759	0.4497
0.1800	0.0358	-0.0732	0.4145
0.1900	0.0351	-0.0708	0.3828
0.2000	0.0344	-0.0685	0.3544
0.2100	0.0337	-0.0663	0.3289
0.2200	0.0331	-0.0643	0.3058
0.2300	0.0324	-0.0624	0.2850
0.2400	0.0318	-0.0607	0.2662
0.2500	0.0312	-0.0591	0.2491
0.2600	0.0306	-0.0575	0.2336
0.2700	0.0301	-0.0561	0.2194
0.2800	0.0295	-0.0547	0.2064

Table 14: (continued)

a	H{M}	dH{M}/da	$d^2H\{M\}/da^2$
0.2900	0.0290	-0.0535	0.1945
0.3000	0.0284	-0.0523	0.1836
0.3100	0.0279	-0.0512	0.1736
0.3200	0.0274	-0.0501	0.1643
0.3300	0.0269	-0.0491	0.1558
0.3400	0.0264	-0.0482	0.1478
0.3500	0.0259	-0.0473	0.1405
0.3600	0.0255	-0.0464	0.1336
0.3700	0.0250	-0.0456	0.1273
0.3800	0.0246	-0.0449	0.1213
0.3900	0.0241	-0.0442	0.1157
0.4000	0.0237	-0.0435	0.1105
0.4100	0.0233	-0.0429	0.1056
0.4200	0.0228	-0.0423	0.1010
0.4300	0.0224	-0.0417	0.0967
0.4400	0.0220	-0.0411	0.0926
0.4500	0.0216	-0.0406	0.0888
0.4600	0.0212	-0.0401	0.0851
0.4700	0.0208	-0.0397	0.0817
0.4800	0.0204	-0.0392	0.0784
0.4900	0.0200	-0.0388	0.0753
0.5000	0.0196	-0.0384	0.0723
0.5100	0.0192	-0.0381	0.0695
0.5200	0.0188	-0.0377	0.0668
0.5300	0.0185	-0.0374	0.0642
0.5400	0.0181	-0.0371	0.0618
0.5500	0.0177	-0.0368	0.0594
0.5600	0.0174	-0.0366	0.0571

Table 14: (continued)

a	H{M}	dH{M}/da	$d^2H\{M\}/da^2$
0.5700	0.0170	-0.0363	0.0549
0.5800	0.0166	-0.0361	0.0528
0.5900	0.0163	-0.0359	0.0507
0.6000	0.0159	-0.0357	0.0487
0.6100	0.0156	-0.0355	0.0467
0.6200	0.0152	-0.0353	0.0448
0.6300	0.0149	-0.0352	0.0430
0.6400	0.0145	-0.0351	0.0412
0.6500	0.0142	-0.0350	0.0394
0.6600	0.0138	-0.0349	0.0376
0.6700	0.0135	-0.0348	0.0359
0.6800	0.0131	-0.0348	0.0341
0.6900	0.0128	-0.0347	0.0324
0.7000	0.0124	-0.0347	0.0307
0.7100	0.0121	-0.0347	0.0289
0.7200	0.0117	-0.0347	0.0272
0.7300	0.0114	-0.0348	0.0254
0.7400	0.0110	-0.0348	0.0235
0.7500	0.0107	-0.0349	0.0217
0.7600	0.0103	-0.0350	0.0197
0.7700	0.0100	-0.0351	0.0177
0.7800	0.0096	-0.0353	0.0156
0.7900	0.0093	-0.0355	0.0134
0.8000	0.0089	-0.0357	0.0110
0.8100	0.0086	-0.0359	0.0084
0.8200	0.0082	-0.0362	0.0057
0.8300	0.0078	-0.0366	0.0027
0.8400	0.0075	-0.0369	-0.0007

Table 14: (continued)

a	H{M}	dH{M}/da	$d^2H\{M\}/da^2$
0.8500	0.0071	-0.0373	-0.0044
0.8600	0.0067	-0.0378	-0.0085
0.8700	0.0063	-0.0384	-0.0133
0.8800	0.0059	-0.0390	-0.0188
0.8900	0.0056	-0.0397	-0.0253
0.9000	0.0052	-0.0405	-0.0331
0.9100	0.0047	-0.0414	-0.0426
0.9200	0.0043	-0.0425	-0.0546
0.9300	0.0039	-0.0438	-0.0701
0.9400	0.0034	-0.0454	-0.0910
0.9500	0.0030	-0.0474	-0.1206
0.9600	0.0025	-0.0498	-0.1657
0.9700	0.0020	-0.0532	-0.2421
0.9800	0.0014	-0.0582	-0.3979
0.9900	0.0008	-0.0671	-0.8759

Vita

Gwendolyn Harville Walton was born in Rogersville, Tennessee on June 28, 1952. She graduated from Rogersville High School in June of 1970. In 1974, she graduated from the University of Tennessee with a B.S. in Mathematics Education. From 1974 to 1977, Ms. Walton worked in Knoxville, Tennessee as a Graduate Teaching Assistant in the Department of Mathematics at the University of Tennessee and as a mathematics teacher at Rule High School. She graduated from the University of Tennessee with a M.S. in Mathematics in 1977.

From 1978 through 1984, Ms. Walton worked as a Research Associate in the Energy Division at Oak Ridge National Laboratory, as an Assistant Professor and Department Head of Business Data Processing at State Technical Institute at Knoxville, and as a Senior Data Systems Programmer for Lockheed Missiles and Space Company, Inc. From 1985 through 1993, Ms. Walton worked for Science Applications International Corporation. She served as a Senior Systems Analyst, leader of the Database Development group, manager of the Systems Applications and Development Division, and Assistant Vice President. In 1993 and 1994, Ms. Walton worked as a Graduate Research Assistant in the Software Quality Research Laboratory of the University of Tennessee. In 1994 and 1995, Ms. Walton worked as a consultant for Software Engineering Technology, Inc.

Ms. Walton was awarded the Doctor of Philosophy degree in Computer Science from the University of Tennessee in May of 1995.