

Copyright © 1981, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

ERRATA

- page 69 In Figure 5.5.1, the original curve was wrongly drawn. In fact, all curves are hardly distinguishable from each other. This can easily be verified by examining Table 5.5.1 on page 70.
- pages 78,79,80 Figures 5.5.8, 5.5.9 and 5.5.10 show the smoothed spectral power density function \hat{g}_s and not simply \hat{g} as stated.

DESIGN AND ANALYSIS OF PROGRAM BEHAVIOR MODELS
FOR THE REPRODUCTION OF WORKING-SET CHARACTERISTICS

by
Newton Fallier

Memorandum No. UCB/ERL M81/59

25 June 1981

ELECTRONICS RESEARCH LABORATORY
College of Engineering
University of California, Berkeley
94720

**Design and Analysis of Program Behavior Models
for the Reproduction of Working-Set Characteristics**

Newton Fallar

Ph.D in Engineering

Dept. of Electrical Engineering
and Computer Sciences



Domenico Ferrari
Chairman of Thesis Committee

Abstract

In paged virtual memory systems the concept of the working-set size of a program in execution, which is informally defined as the variable number of pages required to be resident in the main memory at the various instants of the program's execution time in order for the program to run efficiently, is central to the goal of improving system performance. Generative models of program behavior capable of reproducing working-set size characteristics can be invaluable in the analysis and the tuning of page replacement algorithms and of other aspects of memory policies, since they allow a controlled environment for performing experiments to be constructed.

In this research, a theoretical formulation for the working-set size distribution generated by one of the most common models of program behavior (the Least Recently Used Stack Model) is derived. It is shown that it cannot reproduce the essential characteristics of the distributions generated by

real programs in execution as presented in several empirical studies.

A new model is proposed, based on a Markov chain characterization where states represent working-set sizes. The problems which are encountered with this model when the generation of actual page references is sought are discussed. The recent identification of necessary and sufficient conditions for the generation of a feasible sequence of working-set sizes, i.e., a sequence which can be derived from a string of actual references to page names, suggests the definition of a measure called *potential of decrease* and its incorporation in an n-th order Markov model. It is proved that such a model is capable of generating feasible sequences of working-set sizes.

Comparisons between traces generated by this model and by some of its simpler versions, and those generated by phase-transition models are performed in order to evaluate their ability to reproduce static working-set size characteristics (i.e., distribution descriptors), and dynamic working-set size characteristics. Indices are defined to allow a meaningful intuitive comparison to be performed. Parametric and nonparametric statistics as well as autocorrelation and spectral analysis techniques are also used for this purpose.

The model is shown to perform better under criteria involving static characteristics than under those involving dynamic ones. This is not surprising, since the model is designed to reproduce static characteristics only. The values of the various indices obtained from the simulation using working-set size strings which produce different forms of working-set size density functions should help one choose a model when trace-driven simulation studies of memory policies are to be performed.

Acknowledgements

This research was sponsored by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (Brazilian National Council for Scientific and Technological Development) and by the Universidade Federal do Rio de Janeiro (Federal University of Rio de Janeiro). I want to thank these two institutions for their financial support, without which it would have been impossible to engage in this doctoral program. I also acknowledge NSF Grant MCS80-12900 and Contract no. N00039-80-K-0849 which supported my use of various computers during the development of this research.

To Maria Ester

I would like to thank my advisor, Professor Domenico Ferrari, for providing me not only with the topic of this research but also with the encouragement and support until its conclusion. To Professor Chittoor Ramamoorthy and Professor Sheldon Ross, I thank for kindly accepting to participate in my thesis committee.

I thank the members of the PROGRES group for encouragement, suggestions and advice not only during the development of this research but also during the painful phase of writing and typing this dissertation. In chronological order, I am indebted to Edwin Lau, Makoto Kobayashi, Jehan François Pâris, Özalp Babaoglu, Juan Porcar, Frank Olken and Luis Cabrera.

A special word of thanks goes to Prof. Jean-Paul Jacob who helped me, even before my arrival at Berkeley, with his understanding of the meanders of the U.C. academic life, and shared with me some of his knowledge in order to avoid the pitfalls of the path to the Ph.D. degree.

I would like to thank my friends Ivan da Costa Marques and Ysmar Vianna e Silva Filho, who graduated from Berkeley some years ago, for their encouragement and advice, respectively, in choosing Berkeley for my graduate studies. I have to acknowledge that both were right.

To my parents, I am grateful for the background education they provided me, sometimes with no small sacrifice, and without which it would have been impossible to pursue a Ph.D. degree. To my parents-in-law, for their uncompromising optimism which encouraged me during these five years at Berkeley, I am indebted.

I want to thank my daughter, Maria Clara, who, with her constant request for attention, provided me with a good balance between the private and the academic life.

Finally, the person I wish to thank most and to whom I am deeply indebted is my wife Maria Ester, to whom this thesis is dedicated. I have no doubt that her support, understanding, companionship and love were absolutely essential for me to achieve this degree.

Table of Contents

Chapter 1

Program Behavior Modeling

1.1 Introduction	1
1.2 Construction, Calibration and Validation of a Model	3
1.3 Models of Program Behavior	8
1.4 Purpose of this Work	9

Chapter 2

The Working-Set Size Distribution and the LRU Stack Model

2.1 Introduction	12
2.2 Working-Set Size Distribution Generated by the LRUSM	15
2.3 LRUSM Obtained from WS Size Distributions	20
2.4 Form of the WS Size Density Function	22

Chapter 3

Development of a New Model

3.1 Introduction	24
3.3 The Markov Model of Working-Set Sizes	24
3.3 The Concept of Potential of Decrease	29
3.4 The Potential-of-Decrease Model	38

Chapter 4

Appendix I

Methodology for Model Validation

Appendix II

4.1 Introduction	49
4.2 Defining the Programs to be Modeled	50
4.3 Defining the Scope of the Validation Procedure	51
4.4 Statistical Methods for Comparison	52
4.4.1 Background	52
4.4.2 Comparison of Static Characteristics	54
4.4.3 Comparison of Dynamic Characteristics	57

Chapter 5

Model Evaluation Through Simulation

5.1 Introduction	60
5.2 The Generation of the Original Traces	60
5.3 Models for Comparison	62
5.4 Duration of the Simulation Runs	64
5.5 Results of the Simulation	69

Chapter 6

Conclusions

6.1 Summary	81
6.2 Directions for Further Research	84

Bibliography	88
---------------------	-----------

CHAPTER 1

Program Behavior Modeling

1.1. Introduction

The study of program behavior deals with the characterization of the amounts of computational resources a computing system allocates to a program during its execution in order to carry the task to its correct completion. Program behavior models are built in order to actually reproduce the consumption, or, at least, provide good estimates of the amounts of computing resources.

Many aspects of the behavior of a program can be modeled. Examples of aspects capable of being modeled include usage of certain registers and usage of specific instructions, at a more hardware oriented level, and input/output from/to disk, buffer utilization and memory space required, at a more software or operating system oriented level.

These models can be used for many purposes. For example, before actually building a new system or introducing some modification to an existing one, the impact on performance can be estimated through the usage of a good model. Models constitute an excellent tool for performance prediction.

With the advent of virtual memories, programs were divided into pieces which, in general, are resident in the main memory during only a fraction of the total time required for the program's execution [Denn70]. If the program

is divided physically into equal size pieces, each piece is called a *page* and the system is said to use a *paged virtual memory*. On the other hand, if the program is divided into logical pieces, in general of different sizes, each piece is called a *segment* and the system is said to use *segmented virtual memory*. There are also some systems where segments are paged. These systems can be considered as paged systems for most performance purposes.

Among the types of systems mentioned above, paged systems will be the only ones considered throughout this work. Specifically, the main concern of this work is the modeling of the sequence of references that a program issues to its pages when executing in a paged virtual memory environment

Models of program behavior have been used mainly for the study of page replacement algorithms. More recently, however, program behavior models, with slight modifications, have been successfully applied to the estimation of performance at both extremes of memory hierarchies, i.e., to the study of cache allocation and file migration.

Although the price of memory has dropped considerably in the last few years, memory is not, and probably will never be, a free resource. Therefore, any research aimed at improving our understanding of how a program behaves in order to save memory space while keeping the same performance level seems to be fully justifiable.

1.2. Construction, Calibration and Validation of a Model

Models of program behavior can be classified into three categories: conceptual models, analytic models and generative models. A conceptual model is defined as an informal description of what might be the memory referencing behavior of a program. An analytic model is a mathematical model. In general, it bears little or no resemblance with any physical aspect of the behavior of a program in execution, but, from appropriate assumptions and accurate data obtained from real programs, it can estimate some performance indices. Generative models are, in terms of the type of their output, the closest ones to real programs. They are capable of generating string of page names and, in some cases, actual references to pages. The former can be used in simulation, the latter in measurement. Generative models are implemented by programs (as analytic models can be), but their structure is usually quite different from that of the program being modeled. Although conceptual models can give some insight into the actual structure of the behavior of a program, and analytic models can estimate some of its aspects, generative models are the only ones capable of substituting real programs for the purpose of actual measurement of system performance.

The first step in the construction of a model is the definition of its internal structure. To each internal structure for a model there underlie, to a certain extent, the conceptions of the modeler about the actual structure governing the behavior of the modeled phenomenon.

There is no theory behind the choice of an internal structure for a model. It is, in most cases, the result of careful observation or, sometimes, just pure insight. However, some general guidelines are usually followed by

successful modelers. First of all, the model should be as simple as possible. There is no advantage in using a complicated structure when a simple one yields comparable results. Secondly, the parameters required for the specification of the actual structure should be reasonable in number and easily obtainable, i.e., the calibration of a model should be a reasonably simple task. While too many parameters, in general, complicate the model unnecessarily, parameters obtained through involved procedures are prone to be loaded with errors making their usage, at best, debatable. Finally, the model itself or, at least, its output should be amenable to mathematical or statistical analysis, i.e., it should allow a reasonably easy validation. A model whose accuracy under a certain criterion cannot be verified, though perhaps not useless, should not be viewed as successfully concluding the execution of a modeling procedure.

Accurate knowledge of the phenomenon to be modeled may help considerably. This does not mean, however, that a full understanding of the internal structure of the phenomenon is essential for the modeling task. As a matter of fact, even nowadays, the structure, if there exists one, underlying the behavior of a program, which, when reproduced by a model, would allow this model to represent accurately the behavior of a real program in all situations, is not known. The choice of the basic structure for a model is, therefore, arbitrary and its appropriateness in representing a real-world phenomenon will be evaluated through the process of validation.

Many structures for models of program behavior have been proposed and investigated. With respect to the page referencing patterns, one can mention: random referencing, i.e., no structure at all; references to pages

independent of each other; reference to the next page dependent on the currently referenced page only; reference to a page dependent on the previous reference to that same page (locality); multiple localities; etc. Many other structures can be devised though not without some sacrifice of the guidelines introduced above.

The final step for model building involves the assignment of values to the model's parameters, thus defining its actual structure for a specific case. After the assignment of a set of values to parameters, some test cases should be chosen and the values of performance indices should be checked. If these indices are not within the error bounds specified by the modeler, another set of parameters should be tried. This procedure is called *calibration* and its objective is to eliminate or, at least, reduce structure formulation inaccuracies. The calibration of a model is intrinsically related to its internal structure. If the structure is a simple one, it might be suspected that the number of parameters is small and the calibration is relatively easy. Although an acceptable set of parameter values may not be easily obtainable, a simpler model, in general, facilitates this task.

The ultimate acceptance of a model, however, is in the *validation* phase. This phase involves the determination of how well a model can represent a real-world phenomenon under specific criteria. The validation of a conceptual model is done indirectly. If a page replacement algorithm based on a conceptual model of program behavior, for instance, outperforms algorithms based on other models, this is a good indication of the validity of that conceptual model. On the other hand, for analytic and generative models, since their results can be directly compared with those obtained from real pro-

grams, mathematical and statistical tools can be, and actually are, used in their validation.

Many criteria can be defined for performing the validation of a model. Among the most common ones, one can mention: the average number of page faults, the mean time between page faults (lifetime curve), the mean memory occupancy, the space-time product, the distribution of memory occupancy, the dynamics of memory occupancy, and many others. What is a good criterion for validation depends on the application purposes of the model.

1.3. Models of Program Behavior

The simplest model of program behavior one can devise is the *random* model. The underlying assumption (structure) is that the pages of a program are referenced randomly. This model is used for comparisons only, since it hardly passes any validity tests.

A little more sophisticated is the *independent reference model (IRM)* [Coff73]. To the event of a page being referenced, a specific probability is assigned, and these events are assumed to be independent. The assigned probabilities are, in general, estimated from real program traces. The IRM does not fare very well in validity tests, although some attempts have been made to adjust its coefficients in order to yield better results in specific cases [Bask76]. The IRM, however, is recognized to be too simple to reproduce the behavior of a real program.

Some degree of interdependence among page references was introduced through *Markov models* [Bogo75]. Some other models where interdependence was taken into account were also devised [East75]. The problem, however, seems to be the memoryless property of Markov processes. It does not seem that programs issue memory references which are dependent only on the page being referenced. Thus, in the process of validation, these models, under most criteria, seem to perform poorly.

Better models can be built when the concept of *locality* [Bela68] is used. During any time interval much shorter than the total duration of a program's execution, memory references can usually be observed to be concentrated in small subsets of its pages. Although the degree of concentration may vary from program to program or even during a given program's execution, the presence of locality in most programs seems to be universally recognized. Program locality has been measured, analyzed and modeled [Coff72, Spir72, Bats76, Madi76]. It has also been shown that, in most programs, locality can be increased by the appropriate rearrangement of the program's logic blocks [Hal71, Ferr74, Ferr75, Baer76, Ferr78]. Currently, it seems that no good model of program behavior can overlook the concept of locality.

The most popular generative model capitalizing on the locality concept is the *least recently used stack model* (LRUSM). Based on the page replacement algorithm that bears the same name (LRU) [Matt70], it associates probabilities not directly to pages but to positions in a stack. These positions are filled with page names and the stack is updated at each new reference in order to keep the most recently referenced page on its top. This page is,

therefore, inserted at the top of the stack at the same time as other pages are shifted downwards (away from the top) until the empty slot left by the currently referenced page is filled. Pages beyond this point are left untouched. The LRUSM seems to be one of the best models currently available.

Another very popular model using the concept of locality is the *working-set model* (WSM) [Denn68]. Even though a conceptual model, it has influenced the design of many actual page replacement mechanisms [Morr72] that seem to perform even better than LRU in most cases, and other components (a dispatcher [Rodr73a], a system for the dynamic partitioning of the main memory [Ghan75]) of operating systems for virtual memory machines. Unfortunately, however, there is at present no working-set-based generative model which allows direct comparisons with the LRUSM to be performed.

Observing the execution of a number of large programs, it was noticed that the utilization of a single model, in general, is not sufficient to characterize its behavior. These programs seem to concentrate their references into a relatively small subset of pages during a relatively long period of time (phase), followed by a short period where almost all references are issued to pages not referenced in the recent past (transition) [Denn78a]. This phase-transition behavior is observed in programs like compilers, though not restricted to them.

Assuming the phase-transition behavior as the basic structure, phase-transition models were devised. For the phase behavior (micromodel) any of the above mentioned models may be used. In general, however, the LRUSM

seems to be preferred. For the transition structure (macromodel), a Markov chain, where states are associated with phases, has been proposed and found to perform reasonably well [Denn75].

Although the phase-transition model seems to be the most suitable one for the representation of program behavior, at least for certain programs, the effort to obtain its parameters (calibration) is by no means trivial. The problem stems from the fact that it is extremely difficult to characterize phases by observing a string of page references generated by a real program. If the span of attention is too short, each page might be considered a phase. If too long, the program as a whole will be a single phase. A variety of phases of different sizes can be obtained by varying the span of attention. The difficulty involved in the partition of systems into modules was assessed by Courtois [Cour75], who studied the applicability of techniques used in econometrics to stochastic models of computer systems.

1.4. Purpose of this Work

The working-set model, though a conceptual model, has been shown to provide a good basis for devising not only practical [Morr72,Smit78] but also theoretical [Prie78] page replacement algorithms. For most programs, they outperform all other algorithms under a number of criteria. This fact suggests that the WSM seems to capture many of the intrinsic characteristics of the behavior of a program in execution [Denn78b,Denn80]. Research on the design and analysis of generative models capable of reproducing working-set characteristics under a variety of different criteria is the main purpose of this work.

The working-set size distribution generated by a program behavior model has been the center of attention of many studies [Denn72a,Ienf78,Spir77,Koba79]. In the early 70's, from the observation that working-set sizes generated by a program in execution were asymptotically uncorrelated, it was assumed that they would have a normal distribution [Denn72b]. Subsequent empirical studies [Rodr73b], however, have shown that the property of being asymptotic uncorrelated, though satisfied by working-set size strings generated by real programs, was by no means a guarantee for this assumption. In fact, more recent studies [Dya75,Alan80] have found, in most cases, multimodal working-set size density functions when real programs' traces were measured. Therefore, any program behavior model intended to reproduce working-set characteristics must have the capability of generating multimodal working-set size density functions.

In the search for such models, the first, and trivial, attempt to be made is the use of some previously defined models which have already shown good modeling capabilities, e.g., the LRUSM and the phase-transition model. In Chapter 2 an analytic formula for the working-set size distribution generated by a LRUSM is derived and it is shown that the LRUSM is incapable of generating multimodal distributions. Since the generation of multimodal working-set size density functions is an essential property for a model to have when the correct reproduction of working-set characteristics is sought, the usage of the LRUSM in this context does not seem to be generally justified.

The phase-transition model, however, does not present this problem. Multimodal distributions can be obtained through an appropriate definition

of phases and a correct manipulation of transitions among them. The problem with the phase-transition model, as already mentioned, is in the difficulty of its calibration.

A new approach is attempted based on a Markov model where the size of the working-set is defined as the state. This model, however, is generally incapable of generating actual page references, one of the main goals of this work. Fortunately, the discovery of the properties of feasible working-set strings [Ferr81a] led us to the design of a feasible generative model. The original model, the problem and the new model, besides some calibration considerations, are described in Chapter 3.

The validation of such model involves the definition of suitable criteria for comparing the output of the model (working-set sizes) with those produced by real programs under similar conditions. In Chapter 4 some indices are defined and several statistical criteria are analyzed in terms of their suitability for the above mentioned comparisons.

Besides statistical criteria, a methodology had to be devised for the validation of this model. In Chapter 5 these methods are described and the results obtained by simulation are presented.

Finally, in Chapter 6, a summary of the results obtained is presented together with some considerations about directions for further research.

CHAPTER 2

The Working-Set Size Distribution and the LRU Stack Model

2.1. Introduction

Working set (WS) and least recently used (LRU) are the most important concepts used for the implementation of page replacement algorithms in most current systems. The working-set policy keeps in memory all pages to which references have been issued during the most recent time interval (window). Since the working-set size, i.e., the number of pages to which those references were issued, may vary, the working-set principle is used in the implementation of variable memory page replacement policies. The least-recently-used concept, on the other hand, is used in the implementation of fixed memory page replacement policies. The page to be replaced, as the name indicates, is chosen to be the one to which references have been issued in the farthest past, i.e., the least recently used page. The LRU concept referred to in this section is *local* LRU, i.e., applied to a set of pages of one specific program only. When *global* LRU is used, (i.e., pages of several programs sharing the same stack) the number of pages kept in memory belonging to one specific program may vary. As a whole, however, the LRU policy remains an essentially fixed memory page replacement policy.

More formally, one can say that, if $W_T(t)$ is the set of pages belonging to the working set of a program at time t , i.e., the set of pages to which references have been issued in the interval $(t-T+1, t)$ for $t \geq T$, or $(1, t)$ for $t < T$, then $w_T(t)$, the working-set size at time t , is the cardinality of $W_T(t)$.

Analogously for LRU, if $t(i)$ is the time of the last reference to page i and $d(i)$ is the position counted from the top that page i occupies in the LRU stack (i.e., the *stack distance*), then $d(i) > d(j)$ implies $t(i) < t(j)$, where i and j are page names.

Both of these policies can be adjusted or tuned to certain applications or systems through the appropriate choice of their control parameters. In the working-set case, the control parameter is the length of the window. In the LRU case, it is the size of the program's memory. For WS, the longer the window, the greater the chances of a bigger working-set size. For LRU, the bigger the memory space allocated the greater the chances of a page staying longer in memory.

There are many similarities between these two page replacement policies. Among them, probably the most important one is their exhibiting of the inclusion property with respect to their control parameter. In the case of WS, it can be easily seen that, if a new working set is obtained through a longer window, it will include all pages belonging to the original one. In the case of LRU, if a larger number of pages can be allocated in memory, the current pages in memory will belong to that set as well.

Another striking similarity can be observed if in the LRU stack a time stamp is associated with each page, indicating the time of the last reference to that page. It is not difficult to see that, if a maximum number of pages to be allocated in memory under the LRU policy is defined, there exists a variable window size WS policy which will keep in memory at any given time the same pages. On the other hand, if the allocation of memory follows a WS policy, correspondingly, there is a variable memory LRU policy which keeps in

memory at any given time the same pages as those kept in memory by the WS policy.

The LRU Stack Model (LRUSM) is based on the stack used by the LRU page replacement policy. A probability of a page being referenced is associated not directly with the page but with the position it occupies in the LRU stack. Therefore, if a page is found at a certain instant of time occupying a specific position in the LRU stack, the probability associated with this position is the probability that this page will be referenced next.

Similarly to what the LRU page replacement policy does, the stack is updated at each new reference. All pages occupying positions closer to the top of the stack than that currently being referenced are shifted downwards (away from the top) as the referenced page is placed on the top of the stack.

From the similarities observed between the WS and the LRU page replacement policies, and the fact that the LRUSM is implemented using the LRU concept, it can be conjectured that close relationships exist between the WS characteristics generated by a LRUSM and the parameters of this model. This, in fact, proved to be true. Although iterative formulations were already known for some time [Denn72c] for the relationship between the working-set size distribution generated by a LRUSM and the parameters defining this model, elegant closed forms could be obtained. This will be shown in the following sections.

2.2. Working-Set Size Distribution Generated by the LRUSM

In this section, the following notation will be used:

- a_i probability associated with stack distance i ; a_1 is the probability associated with the top of the stack.
- b_j summation of the j top probabilities.
- $W_T(t)$ working set at a specific time t when a window size T is used.
- $w_T(t)$ cardinality of $W_T(t)$.
- $p(w_T=k)$ probability of W_T at a generic time t having size k ; since, due to the stationarity assumption, i.e., probabilities are independent of time t , the variable t is not important in the calculation and, therefore, is omitted.
- $A_i(z)$ z-transform of $p(w_T)$; this function is defined as:
 $A_i(z) = \sum_{T=i}^{\infty} p(w_T=i)z^T$ for $i > 0$ and $A_i(z) = 0$ for $i \leq 0$.

The probability of a working set of size i with a window of size T being generated by a LRUSM can be expressed by the following difference equation:

$$p(w_T=i) = b_i p(w_{T-1}=i) + (1-b_{i-1}) p(w_{T-1}=i-1) \quad (2.2.1)$$

where it is assumed that $p(w_T=0)=0$ for all $T > 0$, and $p(w_0)=1$, which implies $p(w_0=i)=0$ for $i \neq 0$.

Like a differential equation, though applied to discrete time events, a difference equation establishes relationships where rates of variation for the variables are included in the formulation. In this case, the equation states that the probability of finding a working set of size i at a generic time t using

a window of size T is related to the probabilities of finding at time $t-1$ working sets of sizes i and $i-1$ when a window of size $T-1$ is used. The page referenced at time t , which will belong to $W_T(t)$ but might not be included in $W_{T-1}(t-1)$, has stack distance d . The probability that $d \leq i$ is given by b_i , i.e., by the summation of the probabilities of referencing any of the i pages closest to the top of the stack. In this case, $w_T(t)$ will be equal to $w_{T-1}(t-1)$. The probability that $d > i$ is then $1-b_i$, and this event causes an increase in the working-set size.

The difference equation can be solved through the method of z-transforms. Both sides of equation (2.2.1) are multiplied by z^T and summed:

$$\sum_{T=1}^{\infty} p(w_T=i)z^T = b_i \sum_{T=1}^{\infty} [p(w_{T-1}=i)]z^T + (1-b_{i-1}) \sum_{T=1}^{\infty} [p(w_{T-1}=i-1)]z^T \quad (2.2.2)$$

Since $A_i(z) = \sum_{T=1}^{\infty} p(w_T=i)z^T$, equation (2.2.2) can be written as

$$A_i(z) = b_i z \left[p(w_0=i) + \sum_{T=2}^{\infty} [p(w_{T-1}=i)]z^{T-1} \right] + (1-b_{i-1})z \left[p(w_0=i-1) + \sum_{T=2}^{\infty} [p(w_{T-1}=i-1)]z^{T-1} \right] \quad (2.2.3)$$

But since

$$\sum_{T=2}^{\infty} p(w_{T-1}=i)z^{T-1} = \sum_{T=1}^{\infty} p(w_T=i)z^T = A_i(z)$$

and

$$\sum_{T=2}^{\infty} p(w_{T-1}=i-1)z^{T-1} = \sum_{T=1}^{\infty} p(w_T=i-1)z^T = A_{i-1}(z)$$

equation (2.2.3) can be written as

$$A_i(z) = b_i z [p(w_0=i) + A_i(z)] + (1-b_{i-1})z [p(w_0=i-1) + A_{i-1}(z)]$$

Since $i > 0$ and $p(w_0=i)=0$ for $i \neq 0$, hence

$$A_i(z) = b_i z A_i(z) + (1-b_{i-1})z p(w_0=i-1) + (1-b_{i-1})z A_{i-1}(z) .$$

$$A_i(z) = \frac{(1-b_{i-1})z p(w_0=i-1) + (1-b_{i-1})z A_{i-1}(z)}{1-b_i z} \quad (2.2.4)$$

Knowing that $p(w_0=0)=1$, $b_0=0$ and $A_0(z)=0$, equation (2.2.4) can be expanded recursively as follows:

$$A_i(z) = \frac{(1-b_{i-1})(1-b_{i-2}) \cdots (1-b_2)(1-b_1)z^i}{(1-b_i z)(1-b_{i-1}z) \cdots (1-b_3 z)(1-b_2 z)(1-b_1 z)}$$

or

$$A_i(z) = \frac{z^i \prod_{j=1}^{i-1} (1-b_j)}{\prod_{j=1}^i (1-b_j z)} \quad (2.2.5)$$

In order to calculate the probability of the working-set size as a function of the length of the window T , $A_i(z)$ can be expanded in partial fractions and each fraction subsequently expanded in its corresponding series. Finally, the coefficients of corresponding terms in z are added.

Another, and apparently simpler, method for this calculation is the evaluation in the z complex plan of the integral

$$p(w_T=i) = \frac{-1}{2\pi j} \int_C A_i(z) z^{-i-T} dz$$

where $j=\sqrt{-1}$ and C is a closed contour large enough to enclose all poles.

Fortunately, this integral can be solved by the summation of residues using Cauchy's formula

$$r_a = \left[\frac{1}{(m-1)!} \frac{d^{m-1}}{dz^{m-1}} (z-a)^m g(z) \right]_{z=a}$$

where

r_a is the residue at point a

m is the multiplicity of poles at point a

$g(z)$ is the z -transform $A_i(z)$ multiplied by z^{-i-T}

Assuming that all probabilities associated with positions of the stack in the LRUSM are not null, one has $b_i \neq b_j$ for all $i \neq j$. Under this assumption, no multiple poles exist. This makes the calculation more manageable since the Cauchy's formula can be reduced to

$$r_a = [(z-a) g(z)]_{z=a} \quad (2.2.6)$$

where $g(z) = A_i(z) z^{-i-T}$. Thus, for $w_T=1$ one has

$$p(w_T=1) = \frac{-1}{2\pi j} \int_C A_1(z) z^{-1-T} dz = \frac{-1}{2\pi j} \int_C \frac{z}{1-b_1 z} z^{-1-T} dz$$

Using equation (2.2.6), the general formula for $p(w_T=i)$ can, therefore, be obtained. Hence,

$$p(w_T=i) = \sum_{j=1}^i \left[\left(z - \frac{1}{b_j} \right) A_i(z) z^{-i-T} \right]_{z=\frac{1}{b_j}} \quad (2.2.7)$$

As an example, for $w_T=1$ one has

$$p(w_T=1) = \frac{-1}{2\pi j} \int_C A_1(z) z^{-1-T} dz = \frac{-1}{2\pi j} \int_C \frac{z}{1-b_1 z} z^{-1-T} dz$$

which can be solved using equation (2.2.7). Thus, leaving z out of the calculation and consequently adding one to the value of T , one has:

$$p(w_{T+1}=1) = \left[\frac{-(z - \frac{1}{b_1}) z^{-1-T}}{1-b_1 z} \right]_{z=\frac{1}{b_1}} = \left[\frac{(1-b_1 z) z^{-1-T}}{b_1} \right]_{z=\frac{1}{b_1}} = b_1^T$$

Therefore, $p(w_T)=b_1^{T-1}$. Repeating the same procedure for $i=2$, one obtains:

$$p(w_T=2) = \frac{b_2^{T-1}}{b_2 - b_1} + \frac{b_1^{T-1}}{b_1 - b_2}$$

By repeating the procedure indefinitely, the following closed form can be obtained:

$$p(w_T=t) = \left[\prod_{i=2}^t (1-b_{i-1}) \right] \left[\sum_{j=1}^t \frac{b_j^{T-1}}{\prod_{\substack{k=1 \\ k \neq j}}^t (b_j - b_k)} \right] \quad (2.2.6)$$

where, as usual, $\prod_m^n f = 1$ if $n < m$.

Although this formula was obtained independently by the author, its derivation has already appeared in a different form in [Lenf76]. The final formula has also been presented in [Spir77].

The closed form solution can be easily calculated through a computer program. An inherent limitation, however, should not be disregarded. In order to facilitate the calculation of the formula, it was assumed in the development of the solution that there are no positions in the stack whose probability is zero. Null probabilities will imply divisions by zero in the calculation causing, probably, program interruption. Even with this precaution, the multiplication of a series of small numbers should be executed with care since they may cause underflow problems.

2.3. LRUSM Obtained from WS Size Distributions

In section 2.2 a closed form for the distribution of working-set sizes given the parameters of a LRUSM was obtained. It is reasonable to expect, though this is by no means guaranteed in principle, that some closed form for the inverse problem might exist. This is actually the case, as will be shown in this section.

Equation (2.2.5) is, except for a multiplying constant (see below), a product of functions of the type

$$\frac{(1-b_i)x}{1-b_ix} \quad (2.3.1)$$

This function is the z-transform of the geometric distribution whose density function is given by

$$f_i(T) = b_i^{T-1}(1-b_i) \quad (2.3.2)$$

where $1-b_i$ is defined as the chance of success and T is the number of events until the first success occurs. Its z-transform is given by

$$G_i(z) = \sum_{T=1}^{\infty} b_i^{T-1}(1-b_i)z^T = (1-b_i)z \sum_{T=0}^{\infty} b_i^T z^T = \frac{(1-b_i)z}{(1-b_ix)}$$

Examining equation (2.2.5) closely, it can be seen that the upper limit of the product in the numerator is $t-1$, i.e., one less than that in the denominator. Therefore, except for a multiplying constant $1 - \frac{1}{b_i}$, equation (2.2.5) is a product of t functions of the type shown in (2.3.1), and, thus, the z-transform of the probability density function of the summation of t independent geometrically distributed random variables, as shown in equation (2.3.2). Thus,

$$A_i(z) = \frac{1}{1-b_i} \prod_{j=1}^i G_j(z)$$

Since $G_i(z)$ is the z-transform of a probability density function $f_i(T)$ and $T \geq 0$, then

$$\sum_{T=0}^{\infty} f_i(T) = 1.$$

Knowing that the z-transform of $k \cdot f_i(t)$, where k is a constant, is $k \cdot G_i(z)$, $1-b_i$ can be found as follows:

$$\sum_{T=0}^{\infty} p(w_T=t) = \frac{1}{1-b_i}$$

and, therefore,

$$b_i = 1 - \frac{1}{\sum_{T=0}^{\infty} p(w_T=t)}$$

But $b_i = \sum_{j=1}^i a_j$ implies $a_i = b_i - b_{i-1}$ with $b_0 = 0$. Finally,

$$a_i = \frac{1}{\sum_{T=0}^{\infty} p(w_T=t-1)} - \frac{1}{\sum_{T=0}^{\infty} p(w_T=t)}$$

for $t > 0$ with $\sum_{T=0}^{\infty} p(w_T=0) = 1$.

The actual calculation of this formula is a cumbersome procedure and its exact evaluation is impossible in practice since the calculation involves an infinite number of window sizes. Experiments using a reasonable number of window sizes and a linear interpolation between these values were performed. Approximate stack distance probabilities for simple cases (10-15 pages) were obtained.

This result has more theoretical than practical importance. It does not provide an effective method for calculating stack distance probabilities from working-set size density functions due to the requirement of an infinite number of working-set size density functions. In practice, working-set size distributions are calculated for few values of window sizes. In addition, real programs rarely can be well modeled by an LRUSM. Trying to calculate stack distance probabilities from working-set size distributions obtained from real program traces may lead to such inconsistencies as negative stack distance probabilities being obtained when the method presented above is used.

2.4. Form of the WS Size Density Function

Measurements of working-set size distributions generated by the execution of real programs have shown that, in general, the density function of working-set sizes is multimodal. This characteristic is to be attributed to the execution of a variety of phases which have different working-set sizes. The analysis of the form of the working-set size density function generated by the LRUSM is important if, in the attempt to model the working-set characteristics of real programs, the LRUSM is thought of as a serious candidate for such role.

The direct analysis of equation (2.2.8) seems to be extremely complicated. The problem stems from the fact that an increase of index i by one causes one more term to be appended to the summation, and all other terms have their absolute value increased and their signs changed. In addition, the equation is multiplied by a decreasing factor. Attempts to find simple relationships between $p(w_T=t)$ and $p(w_T=t-1)$ have failed. The analysis of varia-

tions across a family of solutions of difference equations seems absolutely non-trivial.

The analysis of the z-transform function (equation (2.2.5)), however, could give us some insight into the shape of the working-set density function. Assuming $p(w_T=i)$ (equation (2.2.8)) a bidimensional function with independent variables i and T , from section 2.3 it can be seen that, keeping i constant, the distribution of the variable T is that of a summation of i geometrically distributed independent variables with parameters b_i and $1-b_i$. These functions are clearly unimodal since they start from a geometric distribution and the summation of i independent variables approximates a bell-shaped (ultimately a normal) density function by the central limit theorem.

This fact and the knowledge that the average of the summation increases with i suggest that, keeping T constant, the density function of working-set sizes might be a bell-shaped curve as well. The analysis, however, is complicated further by the fact that the density function for each i is multiplied by $1-b_i$.

The evaluation of this function with a variety of different stack distance probabilities has produced unimodal functions in all cases. Although a formal proof is lacking, it seems that a multimodal density function of working-set sizes generated by a LRUSM, if at all possible, is very difficult to obtain. This is one of the reasons why the new models shown in the next chapter have been constructed.

CHAPTER 3

Development of a New Model

3.1. Introduction

As shown in Chapter 2, the use of a LRUSM as the basic program behavior model when the validation criteria include reproduction of a given working-set size distribution is, at best, inappropriate. The modeling of real programs showing multimodal working-set size density functions requires the development of a new model.

Abiding by the guidelines mentioned in Chapter 1, before experimenting with complicated structures in order to build a reasonable model, some simple ones are to be examined. Of course, the essential characteristic that the model we are seeking should present is the capability of generating multimodal working-set size density functions. Due to its relative simplicity and to the vast theory available for its analysis, a Markov model seems to be a natural candidate.

3.2. The Markov Model of Working-Set Sizes

In this Markov model of program behavior there are m states, where m is defined as the maximum working-set size. Each state is identified by an integer i ($1 \leq i \leq m$) and the state at time t is defined by a variable $\underline{w}(t)$. The model is said to be in state i at time t if $\underline{w}(t)=i$. The variable $\underline{w}(t)$ is associated with the value of the program's working-set size $w(t)$ at this same time.

Thus, each state is associated with a specific working-set size instead of with the page being referenced. This is the most obvious choice but by no means the only possible one.

The output of this model is a string of the working-set sizes of a program in execution. The output as such, though appropriate for some studies of memory space allocation, is not sufficient to characterize the output of a generative model. A string of page names is, in fact, the actual output that is sought. The solutions to the problem of generating a string of page names corresponding to a given string of working-set sizes will be analyzed in the next section. Therefore, unless explicitly stated, a string of working-set sizes is to be considered the final output of this model.

The basic assumption underlying a model defined in this way is that, due to the memoryless property of Markov processes, a change in the size of the working set of a program in execution depends (probabilistically) on the previous working-set size only. Even though this is a simplistic assumption, it should not be discarded just because of its simplicity: the validation procedure will determine whether or not a model based on it can be considered an acceptable representation of the program in execution according to the criterion we have adopted.

In order to satisfy the basic necessary condition that characterizes a string of working set sizes, i.e., $|w(t) - w(t+1)| \leq 1$ [Denn72a], this model, as shown in figure 3.2.1, is represented by a Markov chain where transitions to non-neighboring states are forbidden. This diagram is that of a birth-death process. Parameters λ_i , μ_i and κ_i indicate the probability that the next reference to memory will make the working-set size greater than, less than



Figure 3.2.1

or equal to its current size. It is worth noticing that λ_i and μ_i are not necessarily independent of the current state, as is generally assumed in regular M/M/1 queueing systems [Kle175b]. As a matter of fact, it is precisely the presence of different values of λ_i and μ_i for different i 's that permit the generation of multimodal probability density functions in the steady state.

The steady-state probability density function can be calculated from the equilibrium equations. In the steady state, the probability of leaving any state should be equal to the probability of entering the same state. Denoting by P_i the system steady-state probability for state i , one has the following equations:

$$P_1 \lambda_1 = P_2 \mu_2 \quad (3.2.1)$$

$$P_2 \mu_2 + P_2 \lambda_2 = P_3 \mu_3 + P_1 \lambda_1 \quad (3.2.2)$$

and, in the general case:

$$P_i \mu_i + P_i \lambda_i = P_{i+1} \mu_{i+1} + P_{i-1} \lambda_{i-1}$$

From equation (3.2.1) one has

$$P_2 = \frac{\lambda_1}{\mu_2} P_1$$

Substituting equation (3.2.1) into equation (3.2.2):

$$P_2 \lambda_2 = P_2 \mu_2.$$

hence,

$$P_2 = \frac{\lambda_2}{\mu_2} P_1$$

Repeating this process until state m is reached, one observes that:

$$P_i = \frac{\lambda_{i-1}}{\mu_i} P_{i-1} \quad (3.2.3)$$

which yields the general formula:

$$P_i = \frac{\prod_{j=1}^{i-1} \lambda_j}{\prod_{j=2}^i \mu_j} P_1 \quad (3.2.4)$$

But since $\sum_{i=1}^m P_i = 1$, one has:

$$\sum_{i=1}^m \frac{\prod_{j=1}^{i-1} \lambda_j}{\prod_{j=2}^i \mu_j} P_1 = 1$$

assuming, as usual, $\prod_{i=j}^k f(i) = 1$ if $k < j$. Therefore,

$$P_1 = \frac{1}{\sum_{i=1}^m \frac{\prod_{j=1}^{i-1} \lambda_j}{\prod_{j=2}^i \mu_j}}$$

and P_i can be calculated using equations (3.2.3) or (3.2.4).

From equation (3.2.3) it is easily seen that

$$\frac{\lambda_{i-1}}{\mu_i} > 1 \rightarrow P_i > P_{i-1}$$

$$\frac{\lambda_{i-1}}{\mu_i} < 1 \rightarrow P_i < P_{i-1}$$

Therefore, with a correct choice of parameters λ_i and μ_i , any form for the steady-state probability density function can be obtained.

Another problem worth investigating is the possibility of obtaining the transition probabilities, λ_i , μ_i and κ_i ($i=1, \dots, m$), for this Markov model given the vector of steady-state probabilities P . As will now be shown, however, this requires some additional information.

From equation (3.2.3) one can obtain $m-1$ independent equations of the form:

$$\frac{P_i}{P_{i-1}} = c_{i-1}$$

where $c_i = \frac{\lambda_i}{\mu_{i+1}}$. Therefore,

$$\mu_i = \frac{\lambda_{i-1}}{c_{i-1}} \quad (3.2.5)$$

Knowing that for each state the summation of transition probabilities must equal one, m additional independent equations of the type

$$\mu_i + \kappa_i + \lambda_i = 1 \quad (3.2.6)$$

can be obtained. Since there are $3m-2$ unknowns and $2m-1$ independent equations, the system is undetermined and additional information is required for a unique solution to exist.

A natural candidate to provide this information is the page-fault rate, i.e. the rate at which new pages are brought into memory. Since the action of bringing a new page into memory causes in most cases an increase in the working-set size, the page fault rate can be utilized to estimate the parameters λ_i . Ideally, one has to measure the conditional page fault rates $f_i = f(w(t)=i)$, and set $\lambda_i = f_i$.

If the $m-1$ parameters λ_i ($i=1, \dots, m-1$) are given, the system of equations becomes determined and can be solved by calculating the κ_i 's as follows:

$$\kappa_i = 1 - \lambda_i - \frac{\lambda_{i-1}}{c_{i-1}}$$

where $a_i = \frac{P_i}{P_{i-1}}$ and, for convenience, it is assumed $\lambda_0 = 0$ and $c_0 = 1$.

Finally, the μ_i 's are obtained from equations (3.2.8).

3.3. The Concept of Potential of Decrease

The model presented in the previous section has a major drawback. It is not possible to use it as a generative model when actual references to pages have to be generated. The problem is that, among the possible sequences of states \underline{w} generated by this model, there might be some which are unrealizable for some value of T . An example should make this point clear.

In figure 3.3.1, taken partially from [Ferr81a], a possible sequence of states (\underline{w}) is assumed to have been generated by the model defined in section 3.2, due to the fact that $|\underline{w}(t+1) - \underline{w}(t)| \leq 1$. The realization of an actual string of references to page names (\underline{r}) is tried. An attempt at reproducing a

\underline{w}	1 1 2 3 3 4 5 5 5 4 5 6 6 5 4 5 4 3 2 3 2 3 4 4 5
$\underline{r}_{T=3}$	a a b c b d e c b c a f f f d d d d c ?
$pd_{T=3}$	0 0 0 0 0 0 0 1 1 1 2 3 2 1 1 1 1 0 0
$\underline{r}_{T=6}$	a a b c a d e b a d f c e d d b b b b a b f c a e
$pd_{T=6}$	0 0 0 0 0 1 2 2 2 3 3 3 2 2 3 2 1 0 1 0 0 0 1 1

Figure 3.3.1

reference string corresponding to the sequence \underline{w} when $T=6$ ($\underline{r}_{T=6}$) fails due to the impossibility of decreasing the working-set size from 3 to 2 at the fifth reference before the end of the string. Alternative solutions may be tried, but they will always fail at the same place, if not earlier. On the contrary, a reference string can be generated when $T=8$ ($\underline{r}_{T=8}$). In fact, it is possible to show that the sequence \underline{w} of working set sizes cannot be obtained when $T \geq 7$. Therefore, the basic condition that characterizes a working-set size sequence, $|\underline{w}(t) - \underline{w}(t+1)| \leq 1$, though necessary, is by no means sufficient.

Recently, the necessary and sufficient conditions that a working-set size string must satisfy to allow the construction of a corresponding page name string have been identified [Ferr81a]. When the window size T and the maximum number of pages of the program npg are given, the necessary and sufficient conditions for a string of integers to be a feasible working-set size string, i.e. to allow the construction of a corresponding page name string, are the following:

- (i) $0 \leq \underline{w}(t) \leq m$
- (ii) $\underline{w}(1) = 1$
- (iii) $|\underline{w}(t) - \underline{w}(t-1)| \leq 1$ for $t=2, \dots, n$
- (iv) $\sum_{i=0}^{T-1} d(t+i) < \underline{w}(t)$ for $t=1, \dots, n-T+1$

where n is the length of the string, $m = \min(T, npg)$ and

$$d(t) = \begin{cases} 1 & \text{if } w_T(t+1) < w_T(t) \\ 0 & \text{otherwise} \end{cases} \quad (3.3.1)$$

While most of these conditions are self-explanatory, in condition (iv) it should be noticed that the summation accounts for the decreases that occur in the interval $[t, t+T-1]$.

The necessary conditions (i), (ii) and (iii) have been known for a long time [Denn72a]. Condition (iv), however, which sets an upper bound for the number of decreases in the working set size during a period of one window size, makes the set of the four conditions sufficient [Ferr81a]. This allows us not only to identify a feasible working set size string for a specific window size but also to generate a corresponding string of page names. In the sequel, these conditions will be referred to as *fwss-conditions*. If m is given, a string of integers satisfying *fwss-conditions* (i), (ii) and (iii) will be called a *working set string (wss-string)*. If, in addition, the value of T is known and this string also satisfies *fwss-condition* (iv), it will be called a *feasible working set size string (fwss-string)* for this specific window size. Thus, even if not explicitly stated, when referring to a *wss-string* or to a *fwss-string* it is assumed that the values of m for the former and m and T for the latter are known.

Lemma 3.3.1

In a *fwss-string* there can be no decreases while $t < T$, i.e., $\sum_{i=1}^{T-1} d(i) = 0$.

Proof:

From *fwss-condition* (iv), if $t=1$ we have

$$\sum_{i=0}^{T-2} d(i+1) < w(1) .$$

Making $\tau = i+1$ and knowing that $w(1)=1$ we get $\sum_{\tau=1}^{T-1} d(\tau) < 1$. Since $d(t) \geq 0$, it must be $\sum_{\tau=1}^{T-1} d(\tau) = 0$.

q.e.d.

According to *fwss-condition* (iv), if a feasible working set size string is to be generated, the number of decreases in the interval $[t, t+T-1]$ must be smaller than the working set size at time t . The feasibility of a decrease in the working set size at each instant of time is conditioned by the decreases which have taken place in the past and can be interpreted as a potential of decrease. More formally, the *potential of decrease* can be defined as

$$pd(t) = \begin{cases} [w(t-T+2)-1] - \sum_{i=1}^{T-2} d(t-i) & \text{if } t \geq T \\ 0 & \text{otherwise.} \end{cases} \quad (3.3.2)$$

The idea of defining a potential of decrease is to provide at each instant of time t a variable such that, if $pd(t)=0$, then $w(t+1)$ must be greater than or equal to $w(t)$ for all *fwss-strings*. It can be observed that the potential of decrease transfers to the past the information that *fwss-condition* (iv) requires from the future. In its definition, the term $[w(t-T+2)-1]$ accounts for the fact that, even when no decreases occurred during the interval $[t-T+2, t-1]$, $pd(t)$ must be zero when $w(t-T+2)=1$. In other words, according to *fwss-condition* (iv), no decrease should be allowed to take place at time t . If decreases have occurred in this same time interval, they are accounted for by the summation. The summation limit $T-2$ is explained by the fact that the potential of decrease, though calculated at time t , refers

to the working set size at time $t+1$. In Figure 3.3.1, besides the page strings, the potentials of decrease for the working-set string w calculated at each instant of time for $T=8$ ($pd_{T=8}$) and $T=9$ ($pd_{T=9}$) are also shown. The time at which the generation of $r_{T=8}$ becomes infeasible can be easily identified.

Some properties of the potential of decrease will now be presented.

Lemma 3.3.2

The potential of decrease calculated for a wss-string with no decreases in the first T references varies in steps of at most one unit, i.e., $|pd(t)-pd(t-1)| \leq 1$ for all $t > 1$.

Proof:

Case 1: For $t < T$, since by definition $pd(t)=0$ (equation (3.3.2)), the property holds.

Case 2: For $t=T$, $pd(T) = [w(2)-1] - \sum_{i=1}^{T-2} d(T-i)$. From fwss-condition (iii), $|w(2)-w(1)| \leq 1$, and from fwss-condition (ii), $w(1)=1$. This makes either $w(2)-1=0$ or $w(2)-1=1$ since, from fwss-condition (i), $w(2) > 0$. Consequently, when $t=T$, it is either $pd(t)=0$ or $pd(t)=1$ because $d(t)=0$ for $t < T$. But since $pd(t)=0$ for $t < T$, $|pd(t)-pd(t-1)| \leq 1$ holds for $t=T$.

Case 3: For $t > T$,

$$\begin{aligned} pd(t)-pd(t-1) &= \\ [w(t-T+2)-1] - \sum_{i=1}^{T-2} d(t-i) - [w(t-T+1)-1] + \sum_{i=1}^{T-2} d(t-1-i) &= \\ w(t-T+2)-w(t-T+1)-d(t-1)+d(t-T+1) \end{aligned}$$

From fwss-condition (iii), $|w(t-T+2)-w(t-T+1)| \leq 1$ for $t \geq T$. Thus, we must consider two subcases.

Subcase 3a: If $w(t-T+2)-w(t-T+1)=-1$, i.e., a decrease occurred from

time $t-T+1$ to $t-T+2$, then $d(t-T+1)=1$ and, thus, $pd(t)-pd(t-1)=-d(t-1)$. Since $0 \leq d(t) \leq 1$ by definition (equation (3.3.1)), we have $|pd(t)-pd(t-1)| \leq 1$.

Subcase 3b: If $w(t-T+2)-w(t-T+1) \geq 0$, i.e., if no decrease occurred from time $t-T+1$ to $t-T+2$, then $d(t-T+1)=0$.

If $w(t-T+2)-w(t-T+1)=0$, then $pd(t)-pd(t-1)=-d(t-1)$ as in Case 3a.

If $w(t-T+2)-w(t-T+1)=1$, then $pd(t)-pd(t-1)=1-d(t-1)$, and thus either $d(t-1)=0$, hence $pd(t)-pd(t-1)=1$, or $d(t-1)=1$, hence $pd(t)-pd(t-1)=0$.

This completes the proof of the Lemma.

q.e.d.

Theorem 3.3.1

Given a wss-string S of length $t+1$ such that its first t elements constitute a fwss-string for window size T , and such that $pd(t)=0$, S is a fwss-string for window size T if and only if $w(t+1) \geq w(t)$.

Proof:

If $t < T$ the proof is trivial since, by definition, for all t from 1 to $T-1$, $pd(t)=0$ and, by Lemma 3.3.1, a fwss-string has no decreases in the interval $[1, T-1]$. Let $t \geq T$

(1) The condition is necessary

Assume a decrease occurs at time t , i.e., $d(t)=1$ and $w(t+1) < w(t)$. From equation (3.3.2), since $pd(t)=0$, one obtains,

$$\sum_{i=1}^{T-2} d(t-i) = w(t-T+2)-1 \quad (3.3.3)$$

Making $r=t-T+2$ and $j=T-2-i$, and substituting in equation (3.3.3), one has

$$\sum_{j=0}^{T-2} d(\tau+j) = w(\tau) - 1 \quad (3.3.4)$$

Since $d(t) = d(\tau+T-2) = 1$, then $\sum_{j=0}^{T-2} d(\tau+j) = w(\tau)$, which violates fwss-condition (iv).

(2) The condition is sufficient

A similar derivation from equation (3.3.3) yields equation (3.3.4). Since no decrease occurs at time t , then

$$\sum_{j=0}^{T-2} d(\tau+j) = w(\tau) - 1.$$

and hence

$$\sum_{j=0}^{T-2} d(\tau+j) < w(\tau).$$

satisfying fwss-condition (iv).

q.e.d.

Theorem 3.3.2

Given a wss-string S of length $t+1$ such that its first t elements constitute a fwss-string for window size T , and that $pd(t) > 0$, S is a fwss-string for window size T .

Proof:

In this case, $t \geq T$ since $pd(t) = 0$ for $t < T$. A derivation similar to that through which equation (3.3.4) was obtained yields

$$\sum_{j=0}^{T-2} d(\tau+j) < w(\tau) - 1.$$

Whether or not a decrease occurs at time t , the inequality

$$\sum_{j=0}^{T-2} d(\tau+j) \leq w(\tau) - 1$$

is true. Hence,

$$\sum_{j=0}^{T-2} d(\tau+j) < w(\tau)$$

which satisfies fwss-condition (iv).

q.e.d.

Theorem 3.3.3

In a fwss-string $pd(t) \geq 0$ for all t .

Proof:

Let us assume that there is an instant of time t' such that $pd(t') < 0$. It should be kept in mind that, from the definition of the potential of decrease, $pd(t) = 0$ for all $t < T$. Thus, by Theorem 3.3.1, no decrease occurs for $t < T$ and, therefore, Lemma 3.3.2 applies to fwss-strings *a fortiori*. Furthermore, still by Theorem 3.3.1, in a fwss-string if $pd(t) = 0$ then $d(t) = 0$ for all t , i.e., no decreases can occur when the potential of decrease is equal to zero.

Case 1: If $t' < T$, the definition of potential of decrease contradicts the hypothesis.

Case 2: If $t' = T$, from Case 2 of Lemma 3.3.2, when $t = T$ we have either $pd(t) = 0$ or $pd(t) = 1$, which contradicts the hypothesis.

Case 3: Let $t' > T$. Knowing that $pd(t) = 0$ for $t < T$, if $pd(t') < 0$, from Lemma 3.3.2 there must have been at least one instant of time, $t-1$ for instance, such that $pd(t-1) = 0$ and $pd(t) = -1$. Thus, since $pd(t) - pd(t-1) = -1$, from Case 3 of Lemma 3.3.2, either

(i) $w(t-T+2) - w(t-T+1) \leq 0$; in this case, $pd(t) - pd(t-1) = -d(t-1)$ yields $d(t-1) = 1$, which contradicts the hypothesis that, since $pd(t-1) = 0$, no decreases occurred at time $t-1$; or

(ii) $w(t-T+2) - w(t-T+1) = 1$; in this case $pd(t) - pd(t-1) = 1 - d(t-1)$ yields $d(t-1) = 2$, which contradicts the definition of $d(t)$. Therefore, in a fwss-string

$pd(t) \geq 0$ for all t .

q.e.d

From the conclusion of Theorem 3.3.3 one might be inclined to use the condition of negative potential of decrease to identify wss-strings which are not fwss-strings. It should be noticed, however, that, while the condition $pd(t) \geq 0$ for all t is a necessary condition for a fwss-string, it is not sufficient. There are cases where wss-strings which are non-feasible for some value of window T have $pd(t) \geq 0$ for all t . This happens when $w(t-T+2) - w(t-T+1) = 1$, $pd(t-1) = 0$ and $d(t-1) = 1$. In this case, fwss-condition (iv) is violated, though, at time t , $pd(t)$ remains equal to zero. The necessary and sufficient condition for the identification and generation of fwss-strings in terms of the potential of decrease is given in Theorem 3.3.1 since, in a fwss-string, $pd(t) < 0$ never occurs (Theorem 3.3.3), and $pd(t) > 0$ causes no problem (Theorem 3.3.2).

Lemma 3.3.3

The maximum value of the potential of decrease at any instant of time t is the value of the working set size at time t minus one, i.e., $pd(t) \leq w(t) - 1$ for all t .

Proof:

The value of $w(t)$ has a lower bound given by the number of decreases occurring in the interval $[t-T+2, t-1]$ subtracted from $w(t-T+2)$. This lower bound is attained when there are no increases in the working set size during the same time interval. Therefore,

$$w(t) \geq w(t-T+2) - \sum_{i=1}^{T-2} d(t-i)$$

Since

$$pd(t) = [w(t-T+2) - 1] - \sum_{i=1}^{T-2} d(t-i)$$

hence $pd(t) \leq w(t) - 1$.

q.e.d

Lemma 3.3.4

The maximum value assumed by the potential of decrease is the maximum value of the working set size minus one.

Proof:

Trivial from Lemma 3.3.3. If $w(t) = m$, then $pd(t) \leq m - 1$.

q.e.d

3.4. The Potential-of-Decrease Model

A new model that takes into account the feasibility of generating a decrease in the working-set size must have some memory of past decreases so that the potential of decrease before a new working-set size is generated can be computed. In this model, the next state does not depend on the current state only. Hence, the model is not a simple first-order Markov model anymore. As will be shown, it is in fact a Markov model of order $T-1$, where T is the window size.

In a generative version of this model, the potential of decrease can be easily calculated if the times of the latest decreases in the working-set size are kept in a vector. The length of this vector is equal to the maximum working-set size, which is, in most cases, much smaller than the total number of pages of the program.

The new model contains a Markov chain for each value of the potential of decrease. The transition probabilities between any two states are estimated,

keeping track of the potential of decrease at each point in time, from an actual string of working-set sizes generated by the program to be modeled. The generation procedure follows the same steps, using at each reference the set of transition probabilities corresponding to the current value of the potential of decrease.

This model has T (the window size) and npq (the total number of pages of the program) as its basic parameters. The bounding parameter m is calculated as the minimum between T and npq . The model's states are identified by the pair (w, pd) corresponding respectively to the working set size and to the potential of decrease. The fact that the model at time t is in state $S(t)=(i, j)$ implies $w(t)=i$ and $pd(t)=j$ and vice-versa. Variable w may take values from 1 to m while pd is bound by 0 and $m-1$. Transitions between states are governed by probabilities λ_i^j , κ_i^j and μ_i^j corresponding to the chances of the three alternatives $w(t+1)=w(t)+1$, $w(t+1)=w(t)$ and $w(t+1)=w(t)-1$, respectively, for $w(t)=i$ and $pd(t)=j$. Transitions yielding $|w(t+1)-w(t)|>1$ or $w(t+1)=w(t)-1$ when $w=1$ or $pd(t)=0$, or $w(t+1)=w(t)+1$ when $w(t)=m$, are forbidden. Transitions between values of pd are governed by equation (3.3.2) using $w(t-T+2)$ and the number of decreases in the last time interval of length $T-1$.

We now summarize the properties of the potential-of-decrease model, called *pdm-properties* in the sequel:

- (i) $S(t)=(i, j)$ implies $w(t)=i$ and $pd(t)=j$, and vice-versa.
- (ii) $1 \leq w \leq m$ and $0 \leq pd \leq m-1$.
- (iii) $\lambda_i^j = p[w(t+1)=w(t)+1 | w=i, pd=j]$
 $\kappa_i^j = p[w(t+1)=w(t) | w=i, pd=j]$

- $\mu_i^j = p[w(t+1)=w(t)-1 | w=i, pd=j]$
- (iv) $p[w(t+1)=w(t)+k] = 0$ for all integers k such that $|k|>1$.
- (v) $p[w(t+1)=w(t)+1] = 0$ for all t such that $pd(t)=0$.
- (vi) $p[w(t+1)=w(t)-1] = 0$ for all $w(t)=1$.
- (vii) $p[w(t+1)=w(t)+1] = 0$ for all $w(t)=m$.
- (viii) $pd(t) = \begin{cases} [w(t-T+2)-1] - \sum_{i=1}^{T-2} d(t-i) & \text{if } t \geq T \\ 0 & \text{otherwise} \end{cases}$

where

$$d(t) = \begin{cases} 1 & \text{if } w(t+1) < w(t) \\ 0 & \text{otherwise.} \end{cases}$$

It should be noticed that underscored variables $w(t)$, $pd(t)$, and $d(t)$ refer to the states of the model, while $w(t)$, $pd(t)$ and $d(t)$ refer to the characteristics of a working-set string. Though for the purposes of this work the corresponding pairs of variables always have the same values at each instant of time t , it is important bear in mind that they constitute two sets of distinct entities.

Theorem 3.4.1

The sequence of states $w(t)$ for $t>0$ generated by the potential-of-decrease model when T and npq are given, $m = \min(T, npq)$ and when $S(1)=(1,0)$, is a *fwss-string*.

Proof:

The string generated by this model satisfies all four *fwss-conditions*, namely:

- (i) $1 \leq w(t) \leq m$

By *pdm-property* (ii), *fwss-condition* (i) is satisfied.

(ii) $w(1)=1$

Since $S(1)=(1,0)$, by pdm-property (i) fws-condition (ii) is satisfied.

(iii) $|w(t)-w(t-1)| \leq 1$

Directly from pdm-property (iv).

$$(iv) \sum_{i=0}^{T-s} d(t+i) < w(t) \quad \text{for } t=1, \dots, s-T+1$$

The working-set size string $w(1)$ of length 1 is certainly a fws-string for all $T \geq 1$. By induction, if the model generated a fws-string of length t , due to pdm-property (viii) and Theorem 3.3.1, the generation of $w(t+1)$ will cause the string of size $t+1$ to satisfy fws-condition (iv).

q.e.d.

Corollary 3.4.1.

The variables $\underline{w}(t)$, $\underline{pd}(t)$ and $\underline{d}(t)$ of the potential-of-decrease model have the same properties of variables $w(t)$, $pd(t)$ and $d(t)$ of fws-strings.

Proof:

Trivial from Theorem 3.4.1. The potential of decrease model generates fws-strings; $w(t)$, $pd(t)$ and $d(t)$ are associated with the variables $\underline{w}(t)$, $\underline{pd}(t)$ and $\underline{d}(t)$ and take, respectively, the same values for all t .

q.e.d.

The structure of this generative model is depicted in Figure 3.4.1, and will be called a *pd-diagram*. It should be noticed that it is not a first-order Markov chain diagram. Though arrows represent probabilities that can be non-zero, the summation of probabilities assigned to arrows leaving a specific state may be greater than 1. This is due to the fact that in the pd-diagram presented, while values are assigned to the probabilities of an increase, of a decrease or of no change in the working-set size, the variation of the potential of decrease is dependent on events occurred in the latest

time interval of length $T-1$. Thus, there are, in general, six arrows leaving a typical state (i.e. a state not on the boundary of the diagram). Each pair of arrows indicates the probability of an increase, of a decrease or of no change in the working-set size. In each pair, each arrow indicates a feasible change of the potential of decrease. Figure 3.4.2 shows a typical state with pairs (x,y) labeling arrows where $x = \underline{w}(t+1) - \underline{w}(t)$ and $y = \underline{pd}(t+1) - \underline{pd}(t)$. It should be remembered that, from Corollary 3.4.1, the variables \underline{w} and \underline{pd} vary at most by one unit from time t to $t+1$.

Since this model is a $(T-1)$ -st-order Markov model, it can be represented as such, even though its representation requires a large number of states. The information contained in all permutations of working-set size decreases during a period of one window size, which are concisely

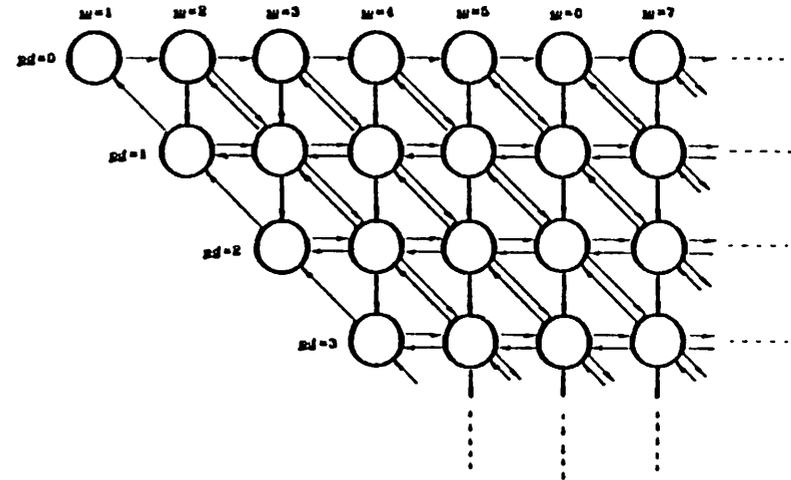


Figure 3.4.1

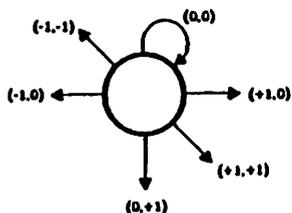


Figure 3.4.2

represented in the pd-diagram, must be explicitly stored in the structure of the chain. The pd-diagram is, therefore, a kind of shorthand notation, since its memory of size $T-1$ is implicit and does not appear in the representation. The pd-diagram allows the same sequence of working-set sizes to be generated while taking into consideration a considerably smaller number of states. Of course, some flexibility is lost, as will be shown below. This means that a state in the pd-diagram accounts for all states in the equivalent $(T-1)$ -st-order Markov model which have the same values of w and pd at any time t . An example should clarify this argument.

Figure 3.4.3 shows the pd-diagram for a potential-of-decrease model where $T=3$ and $npd \geq 3$. The corresponding first-order Markov chain equivalent to this diagram is shown in Table 3.4.1 in matrix form and in Figure 3.4.4 in graph form. The following additional notation has been used in Table 3.4.1:

$dvec(t)$ is the vector of decreases occurred during $[t-T+1, t]$; it keeps the information contained in $[d(t-T+1), d(t-T+2), \dots, d(t-2), d(t-1)]$. This is the information which is embedded in the structure of the first-order

Markov chain and implicit in the pd-diagram;

- indicates an unreachable state;
- indicates an unreachable state in this particular example only.

Since unreachable states are shown in the matrix representation, these states are also shown in the graph representation in Figure 3.4.4 for comparison purposes.

An equivalence between the pd-diagram states and the first-order Markov states can be established. In the Markov state diagram (Figure 3.4.4 and Table 3.4.1), states are defined by a triple $(w, pd, dvec)$ while in the pd-diagram (Figure 3.4.3) states are defined just by the pair (w, pd) . This is

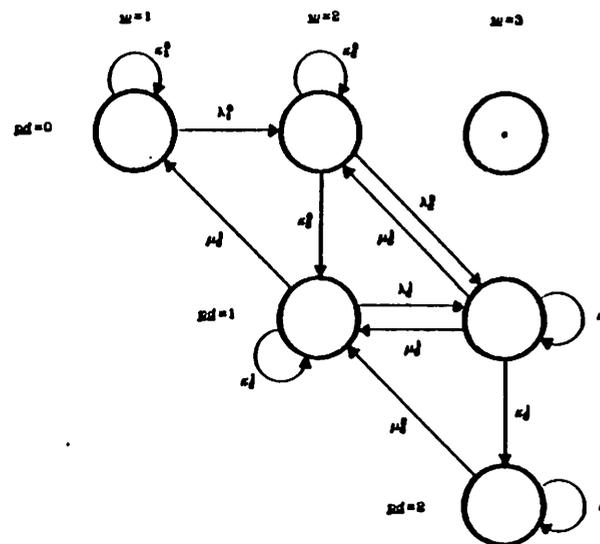


Figure 3.4.3

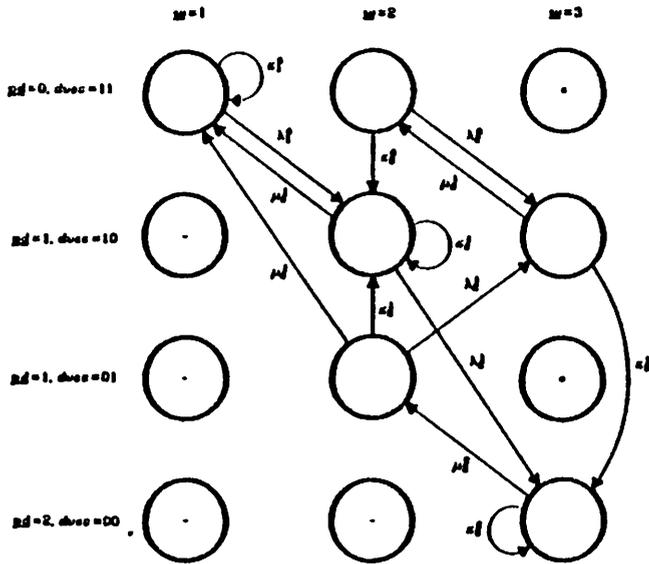


Figure 3.4.4

because the information contained in $dvec$ is used by the pd -model as long as it influences the value of pd . In Figure 3.4.3, for instance, pd -state (2,1) corresponds to states (2,1,10) and (2,1,01) of Figure 3.4.4 or Table 3.4.1. This explains the loss of flexibility caused by the mandatory replications of the same transition probability value for all states which are not differentiated by the pd -model, i.e., those with different values of $dvec$ but the same value of pd .

Comparing Figures 3.4.3 and 3.4.4, it can be seen that the first-order Markov chain representation requires a greater number of states than the

			$w(\ell+1)$											
pd	$dvec$	$w(\ell)$	1	1	1	1	2	2	2	2	3	3	3	3
0	11	1	κ_1^0	0	0	0	0	λ_1^0	0	0	0	0	0	0
1	10	1	-	-	-	-	-	-	-	-	-	-	-	-
1	01	1	-	-	-	-	-	-	-	-	-	-	-	-
2	00	1	-	-	-	-	-	-	-	-	-	-	-	-
0	11	2	0	0	0	0	0	κ_2^0	0	0	0	λ_2^0	0	0
1	10	2	μ_2^1	0	0	0	0	κ_2^1	0	0	0	0	0	λ_2^1
1	01	2	μ_2^1	0	0	0	0	κ_2^1	0	0	0	0	λ_2^1	0
2	00	2	-	-	-	-	-	-	-	-	-	-	-	-
0	11	3	*	*	*	*	*	*	*	*	*	*	*	*
1	10	3	0	0	0	0	0	μ_3^1	0	0	0	0	0	κ_3^1
1	01	3	*	*	*	*	*	*	*	*	*	*	*	*
2	00	3	0	0	0	0	0	0	μ_3^2	0	0	0	0	κ_3^2

Table 3.4.1

potential-of-decrease representation. This difference, however, does not appear so striking since the example was carefully chosen in order to avoid an explosive number of states. In fact, the total number of reachable states in the first-order Markov chain diagram can be shown to be given by $\left(\frac{m+1}{2}\right) \sum_{i=0}^{m-1} \binom{T-1}{i}$, where $m = \min(T, npg)$. This is because, for each value of the working set size, states must be replicated in order to store the information contained in all combinations of decreases, from zero up to $m-1$, which might have occurred during the last time interval of length $T-1$. The term $\frac{m+1}{2}$ accounts for the fact that almost half of the states (those below the main diagonal) are unreachable. Since $T \geq m$, $\sum_{i=0}^{m-1} \binom{T-1}{i} \geq 2^{m-1}$. Thus, the equivalent first-order Markov chain requires a $O(2^{m-1})$ states.

The potential-of-decrease model, though requiring a memory of size $O(m)$ to record the times of the up to $m-1$ decreases that might have taken place during the last time interval of length $T-1$, requires the representation

of $\frac{m(m+1)}{2}$, i.e., $O(m^2)$ states. This can be easily seen because, by Lemma 3.3.3 and Corollary 3.4.1, almost half of the states are unreachable. Since the summation of the parameters λ , κ , and μ for each state must be equal to one, for each reachable state at most two independent parameters should be given. Knowing, however, from pdm-properties (iii) and (iv), that $p[\underline{w}(t+1)=\underline{w}(t)+1|\underline{w}(t)=m]=0$ and $p[\underline{w}(t+1)=\underline{w}(t)-1|pd(t)]=0$, the total number of independent parameters required can be reduced by $2m$ since there are $2m-1$ states where at least one of the transitions is forbidden. Thus, the total number of independent parameters required for the definition of the potential of decrease model is given by $2 \frac{m(m+1)}{2} - 2m = m^2 - m = m(m-1)$. Table 3.4.2 shows a set of independent parameters chosen to define the model presented in Figure 3.4.2.

Since the number of parameters required by this model is very large for practical values of m , an attempt to obtain a simplified model was made before trying to analyze the full model. The essential part required for signaling the infeasibility of a decrease in the working-set size, i.e., the condition of potential of decrease equal to zero, was kept intact. Thus, making $\lambda_j^i = \lambda_i$, $\kappa_j^i = \kappa_i$ and $\mu_j^i = \mu_i$ for all $j > 0$, the number of distinguishable states in the diagram can be much reduced. All states corresponding to the same working

	$pd=0$	$pd=1$	$pd=2$
indep. param.	λ_1^0, λ_2^0	$\lambda_2^1, \mu_2^1, \mu_3^1$	μ_3^2
dep. param.	κ_1^0, κ_2^0	κ_2^1, κ_3^1	κ_3^2

Table 3.4.2

set size having a potential of decrease greater than zero can therefore be lumped together.

The structure for the simplified model is given in Figure 3.4.5. The same interpretation of arrows adopted for Figure 3.4.1 applies also to Figure 3.4.5. The number of states required is $2m-1$ and, using the same argumentation used for the full model, the number of parameters required is found to be $2(2m-1) - (m+2) = 3m-4$, i.e., $O(m)$.

Though the representation of the simplified model can be reduced as shown in Figure 3.4.5, its structure remains the same as that of the full model. Consequently, the pdm-properties are satisfied by this model as well. Actually, the simplification consists of assigning the same value to many parameters and is not the result of a modification in the model's structure. Therefore, Theorem 3.4.1 guarantees that this simplified model, like the full model, generates feasible working set size strings.

The validation of this reduced model, as well as that of the full model, is discussed in Chapter 5.

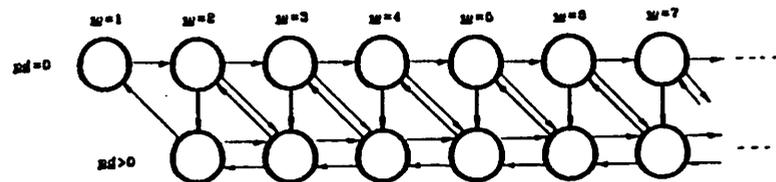


Figure 3.4.5

CHAPTER 4

Methodology for Model Validation

4.1. Introduction

The validation of a model is the verification of the appropriateness of using such a model for the purpose of reproducing a certain real world phenomenon under a specific set of predefined criteria. As already mentioned in previous chapters, the real world phenomenon to be modeled here is the sequence of references a real program in execution issues to virtual memory locations and the main criterion is the reproduction of working-set size characteristics.

In this chapter, the methods to be utilized for the verification of how well the model presented in Chapter 3 reproduces working-set size characteristics are described. First, the target programs, whose behavior is to be reproduced, have to be defined. Secondly, having already chosen the characteristics to be compared, measures for these characteristics have to be specified in order for a quantitative comparison to be feasible. Finally, as a direct result from the comparison of these measures, the criteria to be applied in accepting or rejecting the model in each particular circumstance are selected.

4.2. Defining the Programs to be Modeled

The working-set size distribution functions generated by real programs in execution have been presented in some empirical studies [Drya75, Alan80]. Thus, target programs capable of correctly reproducing such distributions have to be obtained. In order to have a better control over the behavior of the target programs, we decided to use synthetic strings. This approach was particularly interesting in our case, since, as it will be shown in the next chapter, in the simulation process the accuracy of the various indices could be assessed by comparing synthetic strings obtained from the same model using a different string of pseudo-random numbers. For target programs, the phase-transition model was then chosen due to its capability of producing multimodal working-set density functions similar to those presented in the empirical studies mentioned above.

In order to obtain a reasonably representative output without consuming the large amount of computing resources which simulation would require, a series of decisions were made. The target phase-transition model was assumed to have a maximum of thirty pages. The window size utilized was much shorter than a real-world window size in order to match the shorter mean time the model spends in each locality. With this choice, the steady state could be reached without requiring an extremely long simulation run. The values of the parameters defining the actual structure of the phase-transition model were set by a trial and error process so as to obtain three different working-set size density functions: a unimodal, a bimodal and a trimodal function. The working-set size density functions generated by such models and used in the validation procedure are shown in the next chapter.

4.3. Defining the Scope of the Validation Procedure

When a single, very specific model validation criterion, like the correct reproduction of a program's working-set size distribution is being considered, one should bear in mind that many other additional characteristics of the target program may be completely overlooked. Thus, special care must be taken when the model, validated with a certain set of criteria, is to be used as a substitute for the target program. Essential characteristics of the target program may be absent and, in this case, it is said that the model is being used out of its domain of validity. In this work, some problems arising from the influence of these additional characteristics, in particular those having to do with a program's dynamic behavior, were taken into consideration during the validation procedure.

The correct reproduction of the working-set size distribution may be called *static validation*. This simply means that, during the execution of a program, the fraction of instants of time (considered discrete) at which the working set presents a specific size is approximately the same as that found in the model's output. However, nothing is said about the ways the working set reached this specific size. A static procedure is perfect if one is interested only in static criteria, but, in most cases, it is a mistake to be interested in statics only since static criteria are usually incomplete. For instance, if the total number of references to be generated is known, the working-set size may be increased in unit steps and its value kept constant for a number of time instants proportional to the probability specified for this working-set size. The string generated by this procedure would exhibit the given working-set size distribution but can hardly be used as an actual

generative model to reproduce the behavior of a real program. In the validation procedure, therefore, besides the static aspects, we introduced some considerations about the dynamics of the program's behavior; in other words, we performed also a *dynamic validation*.

Static validation involves the comparison of distributions. Techniques to perform this task, known as goodness-of-fit methods, can be borrowed from statistical theory. Some methods, parametric and nonparametric, are described in section 4.4, which includes a discussion of their inherent limitations in the validation of our model. In addition to the well known statistic techniques to be described, an index, which proved useful as an indicator for model validation, is also introduced. As far as dynamic validation is concerned, spectral analysis methods have been utilized. Even though the autocorrelation and power spectrum functions can shed some light on the difficult problem of characterizing dynamic behavior, the cost of their computation led to the definition of a simpler dynamic index.

4.4. Statistical Methods for Comparison

4.4.1. Background

In statistical theory, the comparison of two distributions, the observed distribution and the expected distribution, involves the verification of a set of assumptions, the definition of a *measure of discrepancy (md)*, and the test of a hypothesis H_0 , the *null hypothesis*, which states that the observed distribution was obtained from a population obeying the expected distribution. In the testing procedure, after the measure of discrepancy has been defined,

a value α is chosen as an upper bound for the probability of rejecting H_0 when it is true. The probability α is called the *significance* of the test, and the range of smaller values (lower discrepancy) that md can assume corresponding to the probability $1-\alpha$ define a *confidence interval* for the non-rejection of H_0 . Thus, if the discrepancy is so high as to make md fall in the α -critical region, i.e., outside the α -confidence interval, H_0 is rejected with a α significance.

Measures of discrepancy are defined to evaluate the differences between the observed distribution and the expected distribution. If these measures depend on the type of the distributions being compared, the test is called *parametric*, since the parameters of the distribution should be known in order to allow for a correct inference to be performed. In the case of independence of the measure from the type of the distributions, the test is *distribution-free* or *nonparametric*. Fortunately, many parametric tests are asymptotically distribution-free, and this permits their (careful) utilization in almost any comparison test. For the purposes of this work, parametric and nonparametric tests were performed during the validation procedure.

It is worth mentioning that, unfortunately, the basic assumption which allows us to perform correct hypothesis testing in distribution comparisons, both in the parametric or in the nonparametric case, does not hold for working-set size strings. The requirement that the observed distribution be obtained through measurements which are independent of each other, i.e., events $w(t)=i$ and $w(\tau)=j$ independent for any i and j when $t \neq \tau$, is not satisfied. Specifically, when the values of τ and t are close, for instance, the

constraint stated in *fw*-condition (iii), $|w(t+1)-w(t)| \leq 1$, suggests highly dependent and correlated values. Autocorrelation measurements have shown that this is actually the case. One should however notice that data are asymptotically uncorrelated, i.e., that their correlation seems to tend to zero as their distance in time tends to infinity. These results will be shown in the next chapter.

The high correlation presented by working-set size data restricts, if not invalidates, the use of parametric and nonparametric tests for the purpose of accepting or rejecting a specific model. The measures defined for these tests can be used as indices for relative comparisons when many observed distributions are tested with respect to an expected distribution. Their statistical meaning, however, is debatable.

4.4.2. Comparison of Static Characteristics

Among the goodness-of-fit parametric statistics defined for distribution comparison, the most common and best known is Pearson's statistic defined as

$$P_e = \sum_{i=1}^m \frac{(f_i - np_i)^2}{np_i}$$

where n is the total number of outcomes (the length of the string, in our case), m is the number of possible types of outcomes (the distinct working-set sizes), and f_i and np_i are the number of outcomes of type i for, respectively, the observed distribution and the expected distribution. The parameter p_i is, therefore, the probability of an outcome of type i under the null hypothesis. Under the assumptions of random sampling and n large, the

asymptotic

distribution of P_e is a chi-square with $m-1$ degrees of freedom; the critical region, i.e., the condition for rejecting H_0 , is of the form $P_e > K$, where K is a constant.

A question arises, at this point, about the size of the sample. How small can a sample be chosen so that the asymptotic distribution is still meaningful for inference purposes? Though no simple answer to this question exists, it is generally agreed that, when the sample size is, at least, four or five times the value of m , the approximation is an acceptable one. Since in the case of working-set sizes the sample is not randomly drawn, the number of samples used in the validation procedure (the length of the working-set size string generated by the model) should be much higher than five times m . The problems related to the size of the sample, in our case the duration of the simulation run, will be analyzed in the next chapter.

Another parametric test is the likelihood-ratio test. The ratio Λ is defined as

$$\Lambda = \prod_{i=1}^m \left(\frac{np_i}{f_i} \right)^{f_i}$$

where f_i and p_i have the same meaning as in the case of Pearson's statistic. The asymptotic distribution of $-2 \log \Lambda$ is a chi-square distribution with $m-1$ degrees of freedom under H_0 . Its critical region is again of the form $-2 \log \Lambda > K$. Since the likelihood-ratio test is computationally more complicated than Pearson's test and since, when the null-hypothesis is true, the two tests are equivalent [Lind76], the likelihood-ratio test was not used in this work.

If the statistical test is to be independent of the type of the distributions, the goodness of fit should be checked through a nonparametric procedure using, for instance, the Kolmogorov test. The Kolmogorov statistic is defined as

$$K_0 = \max_{\text{all } t} |F_o(t) - F_e(t)|$$

where $F_o(t)$ and $F_e(t)$ are, respectively, the observed and the expected distributions. The critical region is, again, of the form $K_0 > K$. Though being a distribution-free test, it should be noticed that the Kolmogorov test also requires a random sampling procedure for inference purposes, and independence cannot be obtained in working-set size data.

There are many other nonparametric tests which may be more appealing, due to their making more use of the available data than the Kolmogorov test does. However, since this does not make their statistical power greater than that of the Kolmogorov test [Cono71], no other nonparametric statistic, besides K_0 , was used in this work.

Due to the fact that P_e and K_0 can only be used as indices since no inference guaranteed to be correct can be drawn, another, apparently more intuitive, index was defined. This index has proved to be useful in providing not only a better understanding about the shortcomings of the model but also some subsidies for the calibration procedure, when that procedure was attempted. The *index of mismatches*, as it was called, is defined as

$$Im = \frac{1}{2n} \sum_{i=1}^m |f_i - np_i|$$

It is not difficult to see that Im indicates exactly the fraction of events in the observed string which do not have a counterpart (the same value) in a string

obeying exactly the expected distribution (the expected string). The factor 1/2 accounts for the fact that a mismatch is counted twice, once in the observed string and once in the expected string. The main advantage of this index seems to be its intuitive appeal. In the case of working-set size strings, for instance, without the help of statistical theory, the modeler can judge more easily whether or not a model can be accepted to perform a specific task if it is known that between the model's output and the expected output a 100./m percent of mismatches occur. This index can be used as any other above mentioned statistics for the purposes of inference. A derivation of its approximate distribution under the null hypothesis is presented in Appendix I.

4.4.3. Comparison of Dynamic Characteristics

A popular way of characterizing the dynamics of a time series is through its autocorrelation function. This function calculates, for each value of the lag k , the correlation between the values of the series at times t and $t+k$. For a generic time series x_t , which might be a sequence of working-set sizes, the autocorrelation function [Box78] is defined as

$$\rho_k = \frac{E[(x_t - \mu)(x_{t+k} - \mu)]}{\sqrt{E[(x_t - \mu)^2] E[(x_{t+k} - \mu)^2]}}$$

where $E(x)$ is the expectation of x and $\mu = E(x_t)$. The best estimate for ρ_k is given by

$$\hat{\rho}_k = \frac{1}{n} \sum_{t=1}^{n-k} [(x_t - \hat{\mu})(x_{t+k} - \hat{\mu})]$$

where n is the length of the time series and $\hat{\mu} = \frac{1}{n} \sum_{t=1}^n x_t$. In this work, com-

parisons are made between the autocorrelation functions calculated for the expected string obtained from the target program and the observed string obtained from the model.

The spectral power density function carries essentially the same information as the autocorrelation function, but in a different form [Jenku88]. Examining the power at various frequencies (number of oscillations per unit of time) sometimes makes not only the comparison of dynamic behaviors but also model calibration easier. The spectral power density function is calculated from the autocorrelation function using the formula

$$\hat{g}(f) = 2 \left[1 + 2 \sum_{k=1}^{mk} \hat{\rho}_k \cos 2\pi f k \right]$$

where mk is the maximum lag for which ρ_k is considered to be different from zero, and f is the frequency of oscillations in cycles per unit of time. Thus, the variable f (frequency) assumes values in the interval $[0, 0.5]$. To allow meaningful comparisons, however, the spectrum should be smoothed. In this research, this smoothing was performed by using a Bartlett spectral window ω_k . The smoothed function is then given by

$$\hat{g}_s(f) = 2 \left[1 + 2 \sum_{k=1}^{mk-1} \omega_k \hat{\rho}_k \cos 2\pi f k \right]$$

where $\omega_k = \frac{1-k}{mk}$.

Since the calculation of spectral analysis functions consumes a considerable amount of computing resources, another simpler dynamic indicator was introduced. This index calculates the maximum variation that a time series experiences during a certain time interval. The maximum variation indicator $mv_k(t)$ is defined as

$$mv_k(t) = \max_{i=0}^{k-1} w(t+i) - \min_{i=0}^{k-1} w(t+i)$$

The difference between the distributions of the values of $mv_k(t)$ for $t=1, \dots, n-k+1$ calculated respectively for the observed and for the expected strings can be evaluated by the techniques given for static validation. It is obvious that this indicator carries much less information than any of the ones provided by spectral analysis like $\hat{\rho}_k$ and $\hat{g}_s(f)$, for instance. However, since its computation is very easy and its value easily understood, the maximum variation indicator proved helpful in the comparison of dynamic characteristics.

CHAPTER 5

Model Evaluation Through Simulation

5.1. Introduction

The procedure for the analysis of the program behavior models described in chapter 3 in terms of the appropriate reproduction of working-set characteristics is discussed in this chapter. The original synthetic traces of the programs to be modeled, as discussed in section 4.2, were obtained from three phase-transition models and are described in section 5.2. The working-set characteristics generated by each of these traces were compared with the ones generated by four other traces obtained from four different models. This procedure is discussed in section 5.3. The selection of the length of the simulation runs is presented in section 5.4, while comparisons of static and dynamic characteristics are shown in tables and figures throughout section 5.5.

5.2. The Generation of the Original Traces

The generation of the original traces used in this chapter was done by utilizing a phase-transition model. Parameters for this model were carefully chosen in order to produce traces which, for a specific window size, reproduce working-set size distributions similar to those reported in several empirical studies.

The phase-transition model utilized is composed of one macromodel governing the transitions between localities (phases) and several micromodels governing the page references within localities. The macromodel is a semi-Markov model where transition probabilities between localities may be chosen freely and the number of references within each locality can be independently specified as a fixed number (deterministic) or a random number. In the latter case, the random number can be chosen from either a uniform or a geometric distribution. In all simulations used in this research, however, only geometric distributions were used. This means that the macromodel was actually a Markov model. As far as the micromodel is concerned, independent LRU stack models were defined for each locality, and pages common to multiple localities could be specified. It should be mentioned that different strings of pseudo-random numbers were utilized for the generation of each different sequence of events. Thus, the macromodel and each micromodel are governed by different strings of pseudo-random numbers. Programs were implemented in Pascal, and a linear congruential pseudo-random number generation routine was used.

Parameters for the phase-transition model were chosen so as to reproduce three different working-set size density functions. A one-phase model was defined for the reproduction of a unimodal working-set size density function. Two-phase and three-phase models reproduce bimodal and trimodal working-set size density functions respectively. In Appendix II the parameter values utilized for this generation are given. These three working-set size density functions are called simply the unimodal, the bimodal and the trimodal w.s.s.d.f. in the sequel.

5.3. Models for Comparison

Traces obtained from four different models were compared with the ones generated through the phase-transition model (the *original traces*) as described in section 5.2. These four traces were respectively generated by another phase-transition model, a simplification of the reduced model described in chapter 3, the reduced model and the full model.

Due to the highly correlated data characteristics of working-set size strings, which invalidate attempts of using elementary statistical inference for comparing distributions, another trace generated by the same phase-transition model, but with a different string of pseudo-random numbers, was utilized for control. Its parameters are the same as those described in Appendix II for the original phase-transition model. This model is referred to as *model 0* in the sequel.

Having observed from measurements of the original traces (which obey the phase-transition model) that the number of instants of time when the potential of decrease was equal to zero was extremely small, a model requiring the least number of parameters was experimented with. This has the structure of the reduced model described in Chapter 3, where, for $pd=0$, $\mu_i = \kappa_i = 0$ and $\lambda_i = 1$ ($i=1, \dots, m-1$), and $\mu_m = \lambda_m = 0$, $\kappa_m = 1$. For $pd > 0$, the λ_i 's are estimated from the original trace using the formula

$$\lambda_i = \frac{\text{no. of page faults when } w=i}{\text{total no. of } w=i \text{ cases}}$$

Working-set size probabilities are also estimated from the original trace using the ratio of the number of working sets of a specific size to the total length of the string. Parameters μ_i and κ_i are then calculated from these

probabilities, and the λ_i 's by using equations (3.2.5) and (3.2.6). This model, referred to as *model 1* in the sequel, requires only $2m-2$ independent parameters, where m is the maximum working-set size.

The models referred to as *model 2* and *model 3* in the sequel are, respectively, the reduced and the full model described in Chapter 3. For model 3 the parameters μ_i , κ_i and λ_i are estimated from the original trace measuring the numbers of decrements, of instances of no change and of increments respectively, for each working-set size while taking into consideration the value of the current potential of decrease. More formally, if *down*, *keep* and *up* are defined as

$$\text{down}(t,i,j) = \begin{cases} 1 & \text{if } w(t+1)=w(t)-1, w(t)=i \text{ and } pd(t)=j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{keep}(t,i,j) = \begin{cases} 1 & \text{if } w(t+1)=w(t), w(t)=i \text{ and } pd(t)=j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{up}(t,i,j) = \begin{cases} 1 & \text{if } w(t+1)=w(t)+1, w(t)=i \text{ and } pd(t)=j \\ 0 & \text{otherwise} \end{cases}$$

and $npd(i,j) = \sum_{t=1}^n [\text{down}(t,i,j) + \text{keep}(t,i,j) + \text{up}(t,i,j)]$, then

$$\mu_i(j) = \frac{1}{npd(i,j)} \sum_{t=1}^n \text{down}(t,i,j) \quad (5.3.1)$$

$$\kappa_i(j) = \frac{1}{npd(i,j)} \sum_{t=1}^n \text{keep}(t,i,j) \quad (5.3.2)$$

$$\lambda_i(j) = \frac{1}{npd(i,j)} \sum_{t=1}^n \text{up}(t,i,j) \quad (5.3.3)$$

for $i=1, \dots, m$ and $j=0, 1, \dots, m-1$, where m is the maximum working-set size and n is the total length of the trace.

For model 2 (the reduced model), the μ_i 's, κ_i 's and λ_i 's are estimated respectively by equations (5.3.1), (5.3.2) and (5.3.3) for $pd=0$. For $pd>0$ the estimates are obtained by

$$\mu_i = \frac{1}{npdp(i)} \sum_{j=1}^{m-1} \sum_{t=1}^n \text{down}(t,i,j)$$

$$\kappa_i = \frac{1}{npdp(i)} \sum_{j=1}^{m-1} \sum_{t=1}^n \text{keep}(t,i,j)$$

$$\lambda_i = \frac{1}{npdp(i)} \sum_{j=1}^{m-1} \sum_{t=1}^n \text{up}(t,i,j)$$

where $npdp(i) = \sum_{j=1}^{m-1} npd(i,j)$.

5.4. Duration of the Simulation Runs

In simulation procedures, when estimates for steady-state indices are sought, two problems associated with the definition of the number and length of simulation runs have to be taken into consideration. The first one is the bias in the measured indices. This bias, which is introduced by initial (and/or final) conditions can be reduced by the utilization of start-up procedures [Wils78a, Wils78b] or, simply, by allowing the simulation to run for a long time after reaching the steady state (assumed to exist). There are no fixed rules, however, to determine when the steady state is attained [Emsh70, Bobi76]. Many heuristic conditions have been suggested throughout the simulation literature [Emsh70, Fish73, Bobi76, Fish78] to characterize, basically, when the measured indices no longer change significantly over time.

The second problem deals with the determination of procedures to reduce the variance of the measured indices' mean in order to have a reasonably narrow confidence interval for this value. In most cases, during a

simulation run, the mean of indices measured over time are so correlated that, unless very sophisticated methods are used [Heid81], statistical inference can be drawn only when distinct runs (assumed independent) are utilized. There are many methods designed to produce variance reduction for the mean of measured indices [Fish73, Klei74, Klei75a, Fish78, Prit79] and several of them have been presented in computer performance evaluation books [Ferr78, Koba79, Saue81].

One of the simplest methods for variance reduction, though expensive, is that of multiple replications. It consists of obtaining the indices of interest from distinct simulation runs starting from the same initial state but using different strings of pseudo-random numbers. When there is stochastic convergence [Hogg78] and the simulation runs are independent of each other the sample variance of the mean value of these indices can be estimated through a statistical procedure. In the irreducible finite-state Markov chain underlying the models defined in Chapter 3 all states are positive recurrent and aperiodic, therefore, ergodic [Ross80]. Due to these properties, when the length of the string tends to infinity, the probability that the system is in a specific state exists and is independent of the initial state i.e., stochastic convergence for these probabilities is guaranteed [Isaa78]. Since our discrepancy indices are based on these probabilities, the greater the number of index values obtained through successive replications, the more their estimated variance can be reduced. The replication process can then be stopped, for instance, when the confidence interval for the mean of the measured indices becomes smaller than a previously established value.

Let k be the number of replications, p be the index to be estimated, p_i be the estimate of p for each replication i , \bar{p} be the mean of the p_i 's ($\bar{p} = \sum_{i=1}^k p_i$), and S be the square root of the sample variance [Jnd78, Hogg78], given by

$$S^2 = \frac{1}{k} \sum_{i=1}^k (p_i - \bar{p})^2.$$

The statistical procedure is based on the fact that, if the replications are independent of each other then the variable

$$z = \sqrt{k-1} \frac{\bar{p} - p}{S} \quad (5.4.1)$$

has a Student's t -distribution with $k-1$ degrees of freedom. It should be noticed that S^2 could have been defined as the unbiased estimator of σ^2 (population variance), i.e., $S^2 = \frac{1}{k-1} \sum_{i=1}^k (p_i - \bar{p})^2$. In this case, the variable x would have been defined as $x = \sqrt{k} \frac{\bar{p} - p}{S}$ which yields the same values for x as the ones obtained from equation (5.4.1).

An a two-sided confidence interval for the index p is then defined as

$$\bar{p} - \frac{S}{\sqrt{k-1}} t_{k-1, \frac{\alpha}{2}} \leq p \leq \bar{p} + \frac{S}{\sqrt{k-1}} t_{k-1, 1-\frac{\alpha}{2}}$$

where $t_{k-1, \frac{\alpha}{2}}$ is obtained from the Student's t -distribution with $k-1$ degrees of freedom. If a random variable y has a Student's t -distribution with $k-1$ degrees of freedom, then $\text{prob}(y < t_{k-1, \frac{\alpha}{2}}) = \frac{\alpha}{2}$. If a maximum length l is specified for this confidence interval, then the sample variance should be reduced until

$$S \leq \frac{\sqrt{k-1} \, l}{2t_{k-1, 1-\frac{\alpha}{2}}}$$

When comparing distributions, however, we want to be sure that the difference between them is not too big. This explains why the critical values for all measures of discrepancy used ($P\bar{e}$, K_0 or Im) are given by values above an upper limit which should not be exceeded if H_0 is not to be rejected. But the null hypothesis is never to be rejected if both distributions are too close. Thus, in our case, an α one-sided confidence interval is preferred and it is defined by

$$p \leq \bar{p} + \frac{S}{\sqrt{k-1}} t_{k-1, 1-\alpha}$$

In this case, if a maximum length l is specified for this confidence interval, the sample variance should be reduced until

$$S \leq \frac{\sqrt{k-1} \, l}{t_{k-1, 1-\alpha}}$$

Independently of which type of confidence interval is used, in a simulation procedure using the method of replications, the reduction of the variance of the indices' mean is performed by increasing the number k of replications.

As shown in section 5.3, parameters for models 1, 2 and 3 were estimated from an original trace generated by a phase-transition model. Replications of this original trace (model 0) were performed in order to evaluate the variance of the working-set sizes generated and, ultimately, select the appropriate length for this trace. From a trace of length 50000 references, the sample variance calculated for measured working-set sizes (at least for the peak values) were found to be within plus or minus 2% of their mean. This was considered reasonable for our purposes and this length

was adopted for the original traces.

The index selected to determine the length of the simulation and the number of replications was Im . This index adds the absolute values of the linear differences between distributions (see Chapter 4) and was calculated cumulatively for each simulation run. Due to the fact that our study is a relative comparison between models, the bias presented by Im is not a main concern as long as it is small and the length of the simulation runs is the same for all models. Anyway, for each run, Im was calculated at intervals of 5000 references and, for the adopted run length of 50000 references, at least the last three measurements were found to be within 0.02, i.e., differing by less than 2% of mismatches. For confidence intervals, a 95% level was chosen, and their total length was made equal to $0.2\bar{p}$ where, in our case, \bar{p} is the average of index Im calculated for distinct replications of the simulation run. A one-sided confidence interval was used, i.e., there was an interval such that a 95% chance that the correct value of the index Im would be smaller than 1.2 times its sample mean. The replication process could be stopped as soon as the number k of replications became sufficient to satisfy the inequality

$$S \leq \frac{0.2 \bar{p} \sqrt{k-1}}{t_{k-1, 0.95}} \quad (5.4.2)$$

In fact, in all cases, very few replications were needed in order to satisfy inequality (5.4.2).

5.5. Results of the Simulation

In this section, results obtained from the comparison between models 0, 1, 2 and 3, and the original trace when unimodal, bimodal and trimodal working-set density functions (w.s.d.f.) were to be generated are presented. It should be pointed out that, in this section, all functions plotted in figures 5.5.1 through 5.5.7 are discrete functions which only exist for non-negative integer values of their independent variables. In the figures, the points indicating these values were joined by straight lines for readability purposes only.

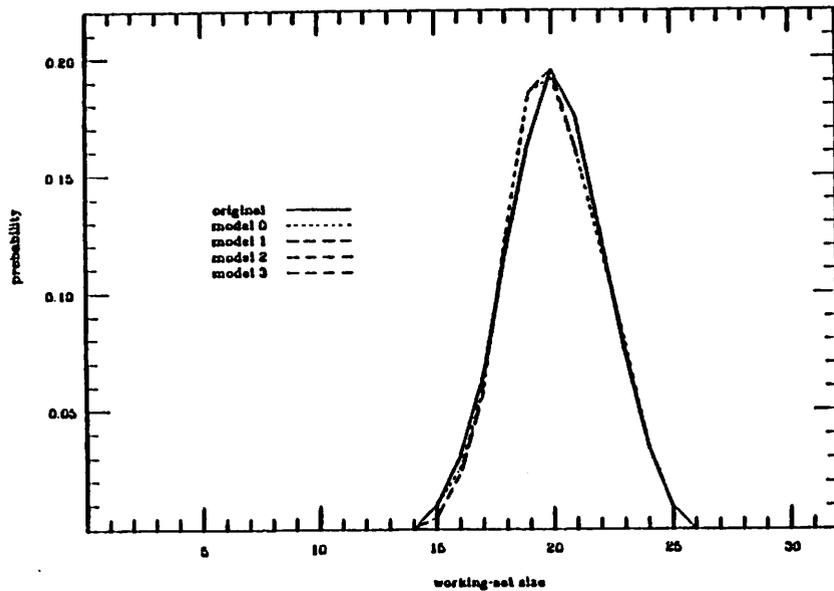


Figure 5.5.1
Unimodal w.s.d.f. generated by different models.

unimodal	P_e	$K_{P_e} (0.95)$	K_0	$K_{K_0} (0.95)$	I_m	$K_{I_m} (0.95)$
model 0	279	42.6	0.018	0.0081	0.027	0.0000
model 1	408	42.6	0.019	0.0081	0.033	0.0000
model 2	522	42.6	0.018	0.0081	0.030	0.0090
model 3	503	42.6	0.017	0.0081	0.025	0.0090

Table 5.5.1
Indices obtained from the generation of an unimodal w.s.d.f.

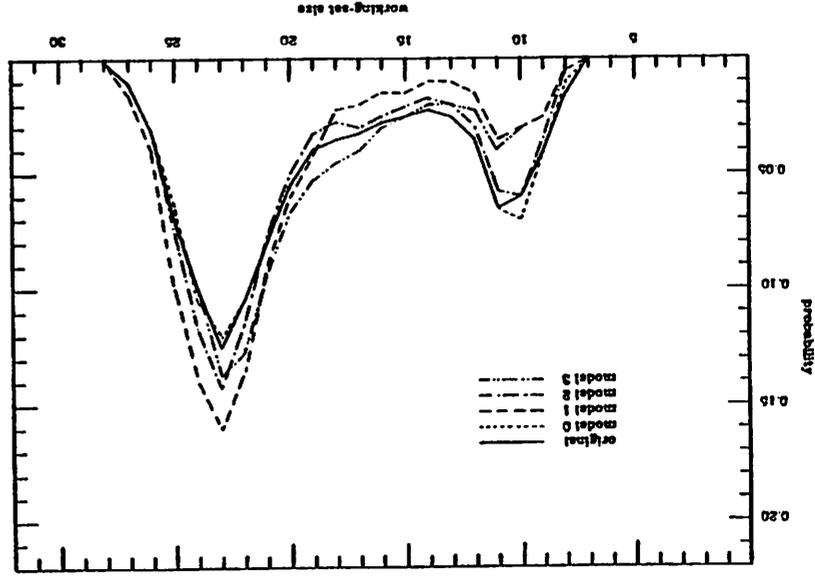
Figures 5.5.1, 5.5.2 and 5.5.3 show the working-set size density functions obtained from the strings generated by the different models when unimodal, bimodal and trimodal working-set size density functions were to be modeled. It is clearly seen in these figures that, for a given, fixed string length (50000 references) the simpler the distribution (i.e., the lower the number of modes), the better the fit. Tables 5.5.1, 5.5.2 and 5.5.3 show a summary of results obtained from the comparison of working-set sizes generated by the (original) phase-transition model and those generated by models 0, 1, 2 and 3 when unimodal, bimodal and trimodal working-set size density functions were to be generated. The upper bounds for K in a 5% significance test under the null hypothesis (H_0), which states that both distributions are equal, are also given in the Tables for comparison purposes only. The values K_{P_e} and K_{K_0} were taken from standard statistics tables [Lind76]. The value K_{I_m} was calculated by an approximate formula derived in Appendix I. It can be observed that the high correlation between successive values of working-set sizes causes, under H_0 , the rejection even of the original model when run with a different string of pseudo-random numbers. It can also be seen that, for the unimodal case, there is no need to use the full model (model 3) since models 1 or 2 yield comparable results. Though model 2 has shown a good performance in reproducing the chosen bimodal w.s.d.f. (Table 5.5.2), in general,

as the complexity of the distribution increases, the accuracy of these simplified models becomes less and less acceptable (Table 5.5.3), and a model with more parameters must be resorted to.

Table 5.5.2
Indices obtained from the generation of a bimodal w.s.s.d.f.

bimodal	P_2	K_{P_2} (0.95)	K_0	K_{P_0} (0.95)	f_m	K_{f_m} (0.95)
model 0	370	42.8	0.007	0.0061	0.021	0.0111
model 1	7212	42.8	0.171	0.0061	0.171	0.0111
model 2	650	42.8	0.052	0.0061	0.052	0.0111
model 3	3344	42.8	0.101	0.0061	0.104	0.0111

Figure 5.5.2
Bimodal w.s.s.d.f. generated by different models.

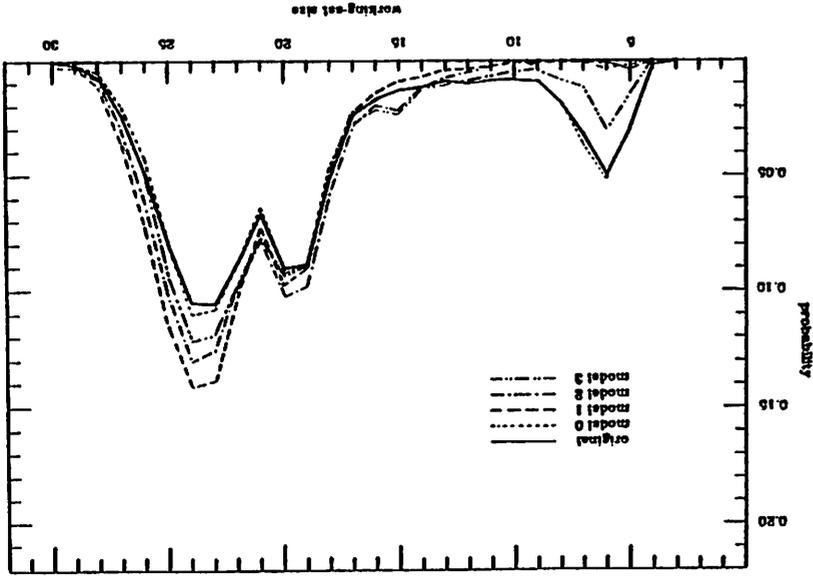


As explained in chapter 4, dynamic characteristics were also taken into consideration. Tables 5.5.4, 5.5.5 and 5.5.8 show the values of f_m for the comparison of the distribution of the maximum variation indicator ($mv(t)$) obtained from the original trace and those obtained from the output traces

Table 5.5.3
Indices obtained from the generation of a trimodal w.s.s.d.f.

trimodal	P_3	K_{P_3} (0.95)	K_0	K_{P_0} (0.95)	f_m	K_{f_m} (0.95)
model 0	374	42.8	0.011	0.0081	0.024	0.0114
model 1	11188	42.8	0.178	0.0061	0.178	0.0114
model 2	6781	42.8	0.158	0.0081	0.158	0.0114
model 3	3009	42.8	0.081	0.0081	0.088	0.0114

Figure 5.5.3
Trimodal w.s.s.d.f. generated by different models.



of the various models when unimodal, bimodal and trimodal working-set size density functions, respectively, were generated. Different values for the variable T (window size) were used and it could be observed that the degree of mismatches seems to be roughly proportional to the simplicity of the model, to the complexity of the working-set density function to be generated, and to the length of the window (at least in the ranges of values explored in our

unimodal	$T=25$	$T=50$	$T=100$	$T=200$	$T=400$
model 0	0.020	0.021	0.043	0.074	0.088
model 1	0.029	0.080	0.123	0.145	0.139
model 2	0.099	0.178	0.199	0.221	0.188
model 3	0.091	0.182	0.198	0.225	0.228

Table 5.5.4
Indices I_m for the distributions of $mv_T(t)$ obtained from the generation of an unimodal w.s.s.d.f.

bimodal	$T=25$	$T=50$	$T=100$	$T=200$	$T=400$
model 0	0.024	0.033	0.083	0.094	0.102
model 1	0.213	0.298	0.438	0.571	0.633
model 2	0.228	0.343	0.485	0.614	0.662
model 3	0.117	0.127	0.198	0.318	0.424

Table 5.5.5
Indices I_m for the distributions of $mv_T(t)$ obtained from the generation of a bimodal w.s.s.d.f.

trimodal	$T=25$	$T=50$	$T=100$	$T=200$	$T=400$
model 0	0.025	0.038	0.035	0.073	0.123
model 1	0.288	0.334	0.387	0.497	0.647
model 2	0.277	0.345	0.424	0.545	0.723
model 3	0.187	0.199	0.258	0.348	0.454

Table 5.5.6
Indices I_m for the distributions of $mv_T(t)$ obtained from the generation of a trimodal w.s.s.d.f.

experiments). Figure 5.5.4 shows the distribution of mv , which explains why a high degree of mismatches was obtained, for instance, for the trimodal function with window length equal to 50 references. It can be observed that the distributions of maximum variations measured from the strings generated by the various models are narrower and their averages are shifted towards the origin (lower values of mv) when compared to the one obtained from the original trace. The same characteristics were observed to a greater or lesser degree in the unimodal and bimodal cases. These remarks suggest a

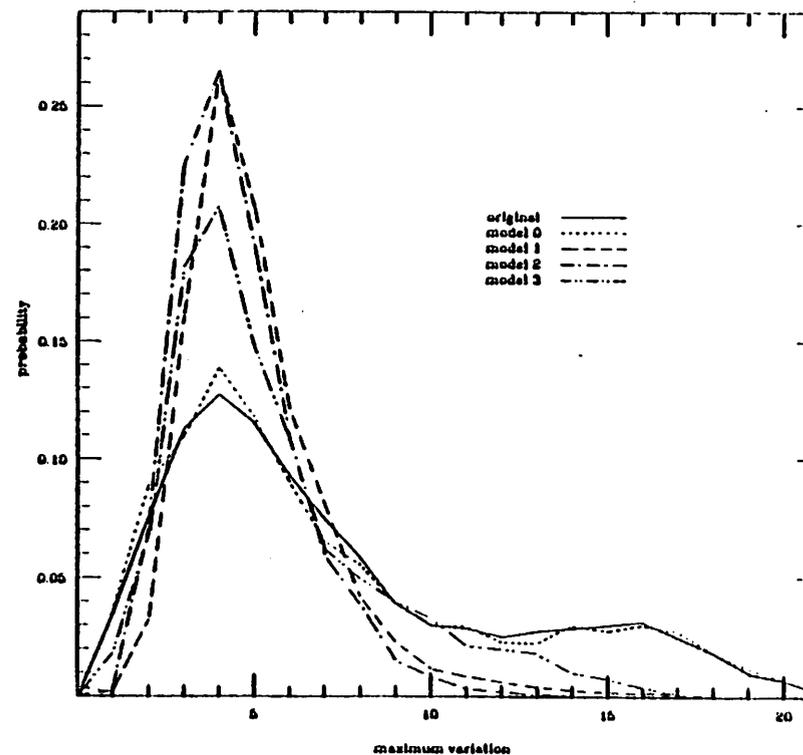


Figure 5.5.4
Distribution of $mv_T(t)$ for $T=50$ obtained from the generation of a trimodal w.s.s.d.f.

higher correlation between successive values of working-set sizes generated by such models since there is a higher chance of correctly predicting the maximum variation.

This conjecture is confirmed by the autocorrelation functions plotted in figures 5.5.5, 5.5.6 and 5.5.7. In fact, in most cases, models 1, 2 and 3 present much higher autocorrelations between working-set sizes generated for higher values of the lag k than the ones presented by the original trace. The same conclusion can be drawn from the examination of figures 5.5.8,

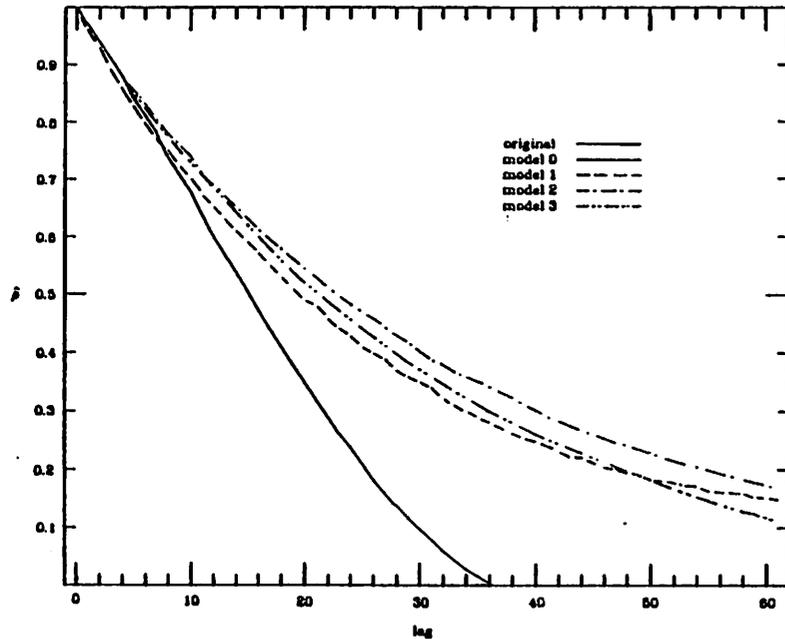


Figure 5.5.5
*Autocorrelation function obtained
from the generation of an unimodal w.s.s.d.f.*

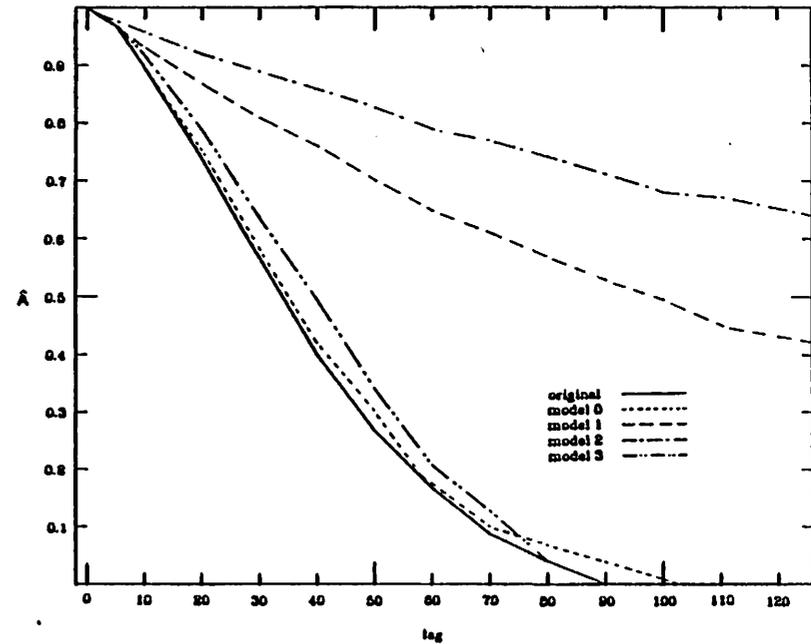


Figure 5.5.6
*Autocorrelation function obtained
from the generation of a bimodal w.s.s.d.f.*

5.5.9 and 5.5.10, where the spectral power density functions for lower frequencies (period greater than 10 units of time) calculated by using the first 100 autocorrelation coefficients are plotted. The power density calculated for frequencies above 0.1 were found to be the same (negligible) for all traces including the original one. In these figures it can be seen that for models 1, 2 and 3 there is a slightly higher concentration of power in the lower frequencies of the spectrum. This, translated in intuitive terms, shows that the proposed models generate traces which produce slower variations of working-set sizes than those present in the original traces.

The models analyzed above, especially model 3, are able to reproduce working-set size density functions reasonably accurately. However, the reproduction of dynamic working-set characteristics does not seem to be as good as the static ones. When such models should be used to produce page reference strings for memory allocation studies is a question that does not have a simple answer. If a reasonable reproduction of the working-set density function must be achieved, we would be inclined to recommend the use of one of these models. However, when the correct dynamic behavior of a specific program or set of programs is also to be reproduced, the case

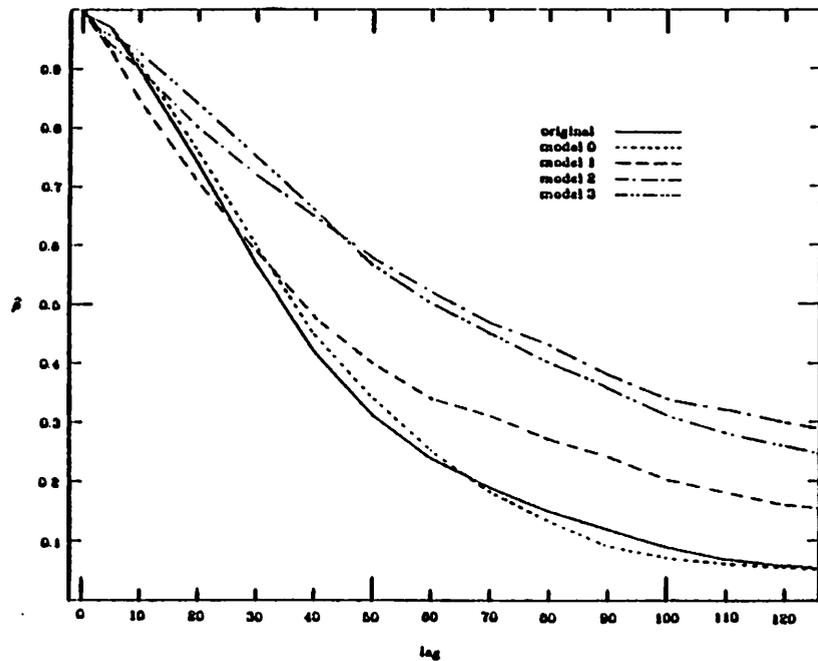


Figure 5.5.7
*Autocorrelation function obtained
from the generation of a trimodal w.s.s.d.f.*

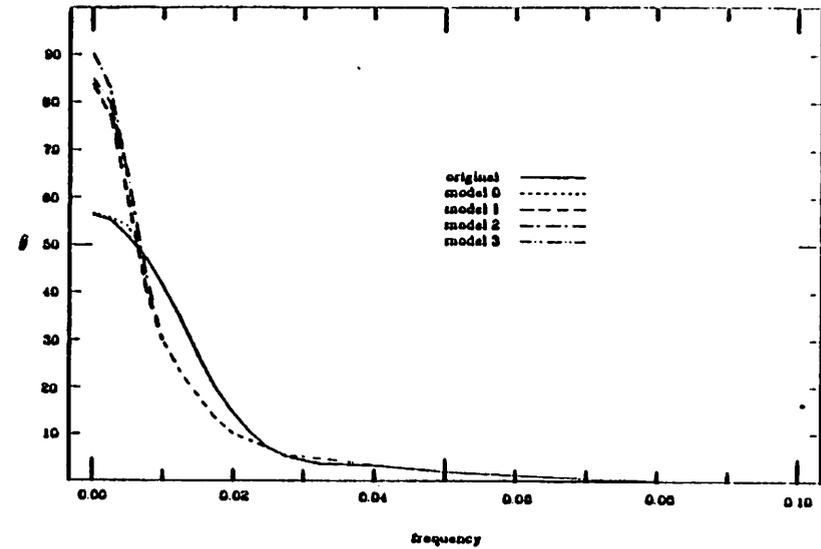


Figure 5.5.8
*Lower part of the spectral power density function
obtained from the generation of an unimodal w.s.s.d.f.*

should be carefully studied before adopting one of these models. Some suggestions on how these models can be modified in order to improve their dynamic behavior is given in the next chapter.

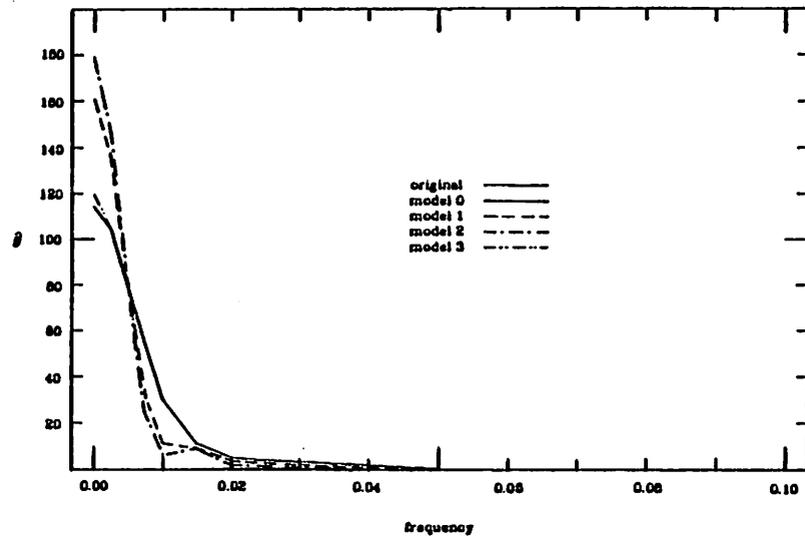


Figure 6.5.9

*Lower part of the spectral power density function
obtained from the generation of a bimodal w.s.d.f.*

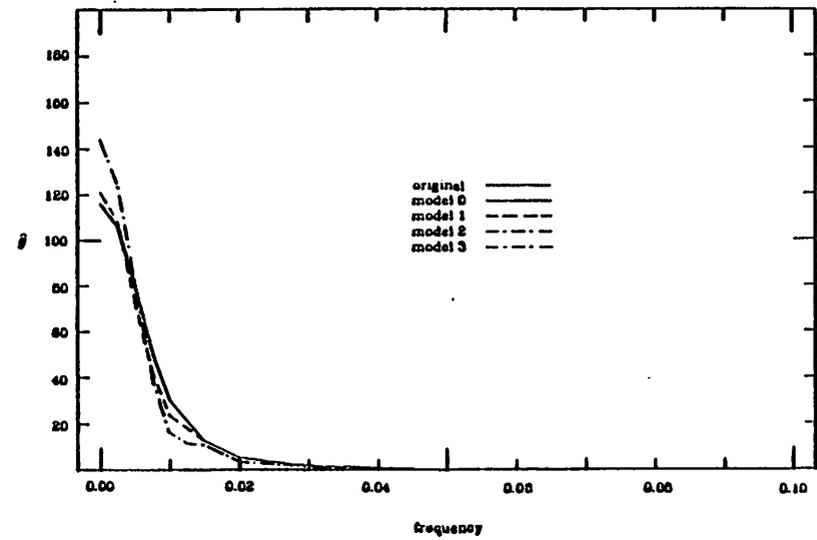


Figure 6.5.10

*Lower part of the spectral power density function
obtained from the generation of a trimodal w.s.d.f.*

CHAPTER 6

Conclusions

6.1. Summary

Despite the constantly dropping price of computer memory, the fact that it will never become a free resource seems to fully justify any research aimed at improving the understanding of how a program behaves in order to save memory space while keeping a desired performance level. Thus, this research was aimed at developing a new program behavior model capable of reproducing some of the working-set characteristics generated by real programs in an easier and/or better way than those provided by the currently available models.

An introduction to the field of program behavior modeling was provided in Chapter 1. Some commonly used models, as well as their advantages and shortcomings, were described. Procedures for the construction, calibration and validation of models were discussed and the main purpose of this research, the construction of models of program behavior capable of reproducing real programs' working-set size distributions, was introduced.

The investigation of working-set size distributions generated by one of the most common models of program behavior, the least recently used stack model (LRUSM), led to the derivation of a closed formula for this distribution through the mathematical technique of z-transforms. Thus, in Chapter 2, relationships between stack distance probabilities of a LRUSM and the

corresponding probabilities of working-set sizes generated by this model were established.

The apparent impossibility of obtaining multimodal density functions for the working set generated by the LRUSM and the difficulty of calibrating a phase-transition model to reproduce a given multimodal working-set size density function led to the development of a new model. The new model was based on a Markov chain where states were associated with working-set sizes instead of with actual page names. The characteristics of this model were presented in Chapter 3.

The possibility that in some cases this model would generate infeasible sequences of working-set sizes, i.e., sequences that cannot be obtained from any actual string of page references, led to the definition of a measure called the *potential of decrease*, which can be calculated as working-set sizes are generated by the model. Still in Chapter 3, it was proved that, by not allowing a decrease in working-set size to occur when the potential of decrease is equal to zero, it is possible to generate feasible working-set size strings.

Based on the concept of potential of decrease, a n-th order Markov model was developed. Due to the high number of parameters required for the definition of such model, a reduced version was devised. An example showing the actual first-order Markov model underlying the n-th order one was also presented for a simple case.

The strings of working-set sizes generated by these models when their parameters were estimated from the strings generated by three phase-transition models producing a unimodal, a bimodal and a trimodal working-

set size density function were compared with them. These shapes of density function were chosen because they are similar to those generated by real programs in execution, as shown by some published empirical studies.

The definition of indices used for comparison purposes, and the analysis of their statistical characteristics when some basic hypotheses are made were discussed in Chapter 4. The actual results of comparisons obtained through simulation were presented in Chapter 5.

As a general conclusion of this research one can state that the models developed, under the testing conditions defined in Chapters 4 and 5, seem to be capable of a reasonably accurate reproduction of working-set static characteristics, i.e., the reproduction of actual working-set size distributions. The reproduction of dynamic characteristics, i.e., the variation of working-set size in time, however, does not seem to be as accurate as that of static ones. This is not surprising, since the model was designed to reproduce static characteristics only. Variations of working-set size produced by the models have shown to be slower than the ones observed in the original traces defined for comparison.

As far as the indices defined for comparison are concerned, some of them (the index of mismatches and the maximum variation, for instance) are intuitive enough to provide guidance on when and whether such models should or not be used as the basis for the generation of actual page references in memory allocation studies.

6.2. Directions for Further Research

The main problem presented by the models defined in this research seems to reside in their inability to reproduce dynamic behavior when somewhat higher frequencies of working-set size variation are present in the string of working-set sizes generated by the program to be modeled. Since our original traces were also generated by a model (phase-transition), it would be interesting to investigate whether the dynamic characteristics of working-set sizes generated by real programs are similar to those generated by this model. If this is actually the case, an attempt to improve the dynamic characteristic of the working-set sizes generated by the proposed models can be made by adjusting the values of λ 's and μ 's. From equation (3.2.3) it is known that the final working-set size density function will not be affected if the ratios between λ 's and μ 's are kept constant. Clever schemes must be devised, however, in order to preserve the feasibility of the values of κ 's when such modifications are performed. Furthermore, despite the higher mobility caused by an increase in the values of the λ 's and of the μ 's, there is no guarantee that with such modifications the model will reproduce better the actual dynamic behavior of real programs. It seems, unfortunately, that a better fit for the dynamic characteristics will hardly be achieved without an increase in the number of parameters required for model definition.

The models developed in this research generate a string of working-set sizes, which should then be processed by an algorithm like the one described in [Ferr81a] if the generation of actual page name strings is sought. The fact that only working-set sizes are manipulated by the model implies that any event occurring in the page name reference string which is not reflected in a

variation of the working-set size is totally overlooked by the model. This happens in the case of the *flat faults* (faults which, being accompanied by the simultaneous departure of a page, do not cause any increase in the working-set size) defined in [Ferr81b]. Since flat faults seem to be important for the correct characterization of dynamic behavior, especially of sequential referencing patterns, an extension of the models presented should be devised in order to take these faults into consideration.

Finally, a validation procedure involving the comparison of results obtained indirectly, i.e., the comparison of performance indices obtained from page replacement algorithms or multiprogramming scheduling policies, for instance, when processing real program's page reference traces and, subsequently, the corresponding model-generated page reference traces, should provide a better measure for the scope of utilization of these models.

Bibliography

- [Alan80] Alanko, T. O., Haikala, I. J. and Kutvonen, P. H. "Methodology and Empirical Results of Program Behavior Measurements," *Proc. of Performance 80, The 7th IFIP W.G.7.3 Int'l Symposium on Computer Performance, Modelling, Measurement and Evaluation*, (May 1980), 55-66.
- [Baer76] Baer, J. L. and Sager, G. "Dynamic Improvement of Locality in Virtual Memory Systems," *IEEE Trans. on Soft. Eng.*, SE-2,1 . (March 1976), 54-62.
- [Bask76] Baskett, F. and Rafil, A. "The AO Inversion Model of Program Paging Behavior," *SLAC PUB-1020, STAN-CS-76-570*, (Oct 1976).
- [Bats76] Batson, A. P. and Madison, A. W. "Measurements of Major Locality Phases in Symbolic Reference Strings," *Proc. of Int'l Symposium on Computer Performance Modelling, Measurement and Evaluation*, (1976), 75-84.
- [Bela66] Delady, L. A. "A Study of Replacement Algorithms for a Virtual-Storage Computer," *IBM Systems J.*, 5,2 . (1966).
- [Bobi76] Bobillier, P. A., Kahan, B. C. and Probst, A. R. *Simulation with GPSS and GPSS V*, Prentice-Hall, 1976.
- [Bogo75] Bogott, R. P. and Franklin, M. A. "Evaluation of Markov Program Models in Virtual Memory Systems," *Software - Practice and Experience*, 5, (1975), 337-346.

- [Box76] Box, G. E. and Jenkins, G. M. *Time Series Analysis: Forecasting and Control*, Revised Ed., Holden-Day, 1976.
- [Brya75] Bryant, P. "Predicting Working Set Sizes," *IBM J. of Res. and Dev.*, 19,3, (May 1975), 221-229.
- [Coff72] Coffman, E. G. Jr. and Ryan, T. A. "A Study of Storage Partitioning Using a Mathematical Model of Locality," *Comm. of the ACM*, 15,3, (March 1972), 185-190.
- [Coff73] Coffman, E. G. Jr. and Denning, P. J. *Operating Systems Theory*, Prentice-Hall, 1973.
- [Cono71] Conover, W. J. *Practical Nonparametric Statistics*, Wiley, 1971.
- [Cour75] Courtois, P. J. "Decomposability, Instability, and Saturation in Multiprogramming Systems," *Comm. of the ACM*, 18,7, (July 1975), 371-377.
- [Denn88] Denning, P. J. "The Working Set Model of Program Behavior," *Comm. of the ACM*, 11,5, (May 1968), 323-333.
- [Denn70] Denning, P. J. "Virtual Memory," *Comp. Surveys*, 2,3, (Sep 1970), 153-189.
- [Denn72a] Denning, P. J. and Schwartz, S. C. "Properties of the Working Set Model" *Comm. of the ACM*, 15,3, (March 1972), 191-198.
- [Denn72b] Denning, P. J. "On Modelling Program Behavior," *SJCC 1972 Conf. Proceedings*, (1972), 937-944.

- [Denn72c] Denning, P. J., Savage, J. E. and Spirn, J. R. "Models for Locality in Program Behavior," *Technical Report 107*, Princeton Univ., Dep. of Eng. Elect., (April 1972).
- [Denn75] Denning, P. J. and Kahan, K. "A Study of Program Locality and Lifetime Functions," *Proc. Fifth SIGOPS*, (Nov 1975), 207-216.
- [Denn78a] Denning, P. J. "Optimal Multiprogrammed Memory Management," in *Current Trends in Programming Methodology, Vol III*, Chandy, K. H. and Yeh, R. T., editors; Prentice-Hall, 1978.
- [Denn78b] Denning, P. J. "Working Sets Then and Now," *Proc. 2nd. Int'l Symposium on Operating System Theory and Practice*, (1978), 115-148.
- [Denn80] Denning, P. J. "Working Sets Past and Present," *IEEE Trans. on Soft. Eng.*, SE-6,1, (Jan 1980), 64-84.
- [East75] Easton, M. C. "Model for Interactive Data Base Reference String," *IBM J. Res. and Dev.*, 19,11, (Nov 1975), 550-556.
- [Ems70] Emshoff, J. R. and Sisson, R. L. *Design and Use of Computer Simulation Models*, Macmillan, 1970.
- [Ferr74] Ferrari, D. "Improving Locality by Critical Working Sets," *Comm. of the ACM*, 17,11, (Nov 74).
- [Ferr75] Ferrari, D. "Tailoring Programs to Models of Program Behavior," *IBM J. of Res. and Dev.*, 19,3, (May 1975), 244-251.

- [Ferr78] Ferrari, D. *Computer Systems Performance Evaluation*, Prentice-Hall, 1978.
- [Ferr81a] Ferrari, D. "A Generative Model of Working Set Dynamics," *Proc. of the Symmetrics Conference on Measurement and Modeling of Computer Systems*, (1981).
- [Ferr81b] Ferrari, D. "Characterization and Reproduction of the Referencing Dynamics of Programs," *UCB/ERL Memo no. M81/82, PROGRES Report no. 81-1*, (April 1981).
- [Fish73] Fishman, G. S. *Concepts and Methods in Discrete Event Digital Simulation*, Wiley, 1973.
- [Fish78] Fishman, G. S. *Principles of Discrete Event Simulation*, Wiley, 1978.
- [Ghan75] Ghanem, M. Z. "Dynamic Partitioning of the Main Memory Using the Working Set Concept," *IBM J. of Res. and Dev.*, 19,5, (Sep 1975), 445-450.
- [Hat71] Hatfield, D. J. and Gerald, J. "Program Restructuring for Virtual Memory," *IBM Systems J.*, 10,3, (1971), 614-620.
- [Heid81] Heidelberger, P. and Welch, P. D. "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations," *Comm. of the ACM*, 24,4, (April 1981).

- [Hogg78] Hogg, R. V. and Craig, A. T. *Introduction to Mathematical Statistics*, 4th Ed., Macmillan, 1978.
- [Isaa78] Isaacson, D. L. and Madsen, R. D. *Markov Chains, Theory and Applications*, Wiley, 1978.
- [Jenk88] Jenkins, G. M. and Watts, D. G. *Spectral Analysis and its Applications*, Holden-Day, 1968.
- [Klei74] Kleijnen, J. P. C. *Statistical Techniques in Simulation: Part I*, Marcel Dekker, 1974.
- [Klei75a] Kleijnen, J. P. C. *Statistical Techniques in Simulation: Part II*, Marcel Dekker, 1975.
- [Klei75b] Kleinrock, L. *Queueing Systems, Volume 1: Theory*, Wiley, 1975.
- [Koba78] Kobayashi, H. *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*, Addison-Wesley, 1978.
- [Koba79] Kobayashi, M. "The Working Set Distribution of the Markov Program Behavior Model," *UCB/ERL Memo no. M79/48*, (July 1979).
- [Lenf78] Lenfant, J. "Comparison of the Working Sets and Bounded Locality Intervals of a Program," *Rapport de recherche no. 41*, Université de Rennes, (1978).
- [Lind76] Lindgren, B. W. *Statistical Theory*, Macmillan, 1978.

- [Madi76] Madison, A. W. and Batson, A. P. "Characteristics of Program Localities," *Comm. of the ACM*, 19,5, (May 1976), 285-294.
- [Matt70] Mattson, R. L., Gecsei, J., Slutz, D. R. and Traiger, I. L. "Evaluation Techniques for Storage Hierarchies," *IBM Systems J.*, 9,2, (1970).
- [Morr72] Morris, J. B. "Demand Paging through Utilization of Working Sets on the MANIAC II," *Comm. of the ACM*, 15,10, (Oct 1972), 887-872.
- [Opde75] Opderbeck, H. and Chu, W. W. "The Renewal Model for Program Behavior," *SIAM J. of Comp.*, 4,3, (Sep 1975), 358-374.
- [Parz60] Parzen, E. *Modern Probability Theory and Its Applications*, Wiley, 1960.
- [Prie76] Prieve, B. G. and Fabry, R. S. "VMIN - An Optimal Variable-Space Page Replacement Algorithm," *Comm. of the ACM*, 19,5, (May 1976), 295-297.
- [Prit79] Pritsker, A. A. B. and Pedgen, C. D. *Introduction to Simulation and SLAM*, Wiley, 1979.
- [Rafi76] Rafii, A. "Empirical and Analytical Studies of Program Reference Behavior," *SLAC Report no. 197*, (July 1976).
- [Rodr73a] Rodriguez-Rosell, J. and Dupay, J.-P. "The Design, Implementation and Evaluation of a Working Set Dispatcher," *Comm. of the ACM*, 16,4, (April 1973).

- [Rodr73b] Rodriguez-Rosell, J. "Empirical Working Set Behavior," *Comm. of the ACM*, 16,9, (Sep 1973).
- [Ross70] Ross, S. *Applied Probability Models with Optimization Applications*, Holden-Day, 1970.
- [Ross80] Ross, S. *Introduction to Probability Models*, 2nd Ed., Academic Press, 1980.
- [Saue81] Sauer, C. H. and Chandy, K. M. *Computer Systems Performance Modeling*, Prentice-Hall, 1981.
- [Smit76] Smith, A. J. "A Modified Working Set Paging Algorithm," *IEEE Trans. on Comp.*, C-25,9, (Sep 1975), 907-914.
- [Spir72] Spirn, J. R. and Denning, P. J. "Experiments with Program Locality," *Proc. AFIPS 1972 FJCC*, AFIPS Press, (1972), 611-621.
- [Spir77] Spirn, J. R. *Program Behavior: Models and Measurements*, Elsevier, 1977.
- [Vant74] Vantilborgh, H. "On the Working Set Size and its Normal Approximation," *Bit*, 14, (1974), 240-251.
- [Wils78a] Wilson, J. R. and Pritsker, A. A. B. "A Survey of Research on the Simulation Startup Problem," *Simulation*, 31, (1978), 55-58.
- [Wils78b] Wilson, J. R. and Pritsker, A. A. B. "A Procedure for Evaluating Startup Policies in Simulation Experiments," *Simulation*, 31, (1978), 79-89.

Appendix I

In this appendix we calculate the asymptotic distribution of the index of mismatches (Im) for the null hypothesis (H_0), i.e., for independently distributed variables.

Let a generic random variable x have a standard normal distribution. Thus,

$$p(x < a) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^a e^{-\frac{x^2}{2}} dx .$$

The random variable $x_i = |z_i|$ thus has a distribution such that

$$p(x_i < a) = \sqrt{\frac{2}{\pi}} \int_0^a e^{-\frac{x_i^2}{2}} dx_i . \quad (1.1)$$

The moments of x_i can be calculated using equation (1.1):

$$E(x_i) = \sqrt{\frac{2}{\pi}} \int_0^{\infty} x_i e^{-\frac{x_i^2}{2}} dx_i = \sqrt{\frac{2}{\pi}} \cdot 1 = \sqrt{\frac{2}{\pi}} . \quad (1.2)$$

$$E(x_i^2) = \sqrt{\frac{2}{\pi}} \int_0^{\infty} x_i^2 e^{-\frac{x_i^2}{2}} dx_i = \sqrt{\frac{2}{\pi}} \cdot \sqrt{\frac{\pi}{2}} = 1 .$$

$$\text{Var}(x_i) = E(x_i^2) - [E(x_i)]^2 = 1 - \frac{2}{\pi} = \frac{\pi-2}{\pi} . \quad (1.3)$$

Since $Im = \frac{1}{2n} \sum_{i=1}^m |f_i - np_i|$, the set of probabilities p_i constitutes a multinomial density function and $x_i = \frac{f_i - np_i}{\sqrt{np_i(1-p_i)}}$ has an asymptotic normal distribution as n increases. Thus, in this case, the definition of Im can be

rewritten as follows:

$$Im = \frac{1}{2n} \sum_{i=1}^m x_i \sqrt{np_i(1-p_i)} = \frac{1}{2\sqrt{n}} \sum_{i=1}^m x_i \sqrt{p_i(1-p_i)}$$

where $x_i = |z_i|$ and z_i has a standard normal distribution $N(0,1)$.

The random variable $y_i = x_i \sqrt{p_i(1-p_i)}$ has a distribution whose mean and variance are given by

$$E(y_i) = \sqrt{p_i(1-p_i)} E(x_i) .$$

$$\text{Var}(y_i) = p_i(1-p_i) \text{Var}(x_i) .$$

If all y_i 's are independently defined and m is large, then Im has an asymptotic normal distribution with mean and variance given by

$$E(Im) = \frac{1}{2\sqrt{n}} \sum_{i=1}^m E(x_i) \sqrt{p_i(1-p_i)} \quad (1.4)$$

$$\text{Var}(Im) = \frac{1}{4n} \sum_{i=1}^m \text{Var}(x_i) p_i(1-p_i) \quad (1.5)$$

But, since only $m-1$ of the y_i 's are independent because $\sum_{i=1}^m p_i = 1$ and

$\sum_{i=1}^m f_i = n$, the variance of Im is, in fact, a little bit bigger. Knowing that

$\sum_{i=1}^m f_i - \sum_{i=1}^m np_i = 0$, one can obtain

$$f_i - np_i = \sum_{\substack{j=1 \\ j \neq i}}^m np_j - f_j . \quad (1.6)$$

However, by introducing absolute values, the number of values of $|f_i - np_i|$ which satisfy (1.6) when only the set of absolute values $|f_j - np_j|$ with $j \neq i$ is known grows with m (the interested reader may verify this by him/herself). Due to this fact, one is inclined to believe that $|f_i - np_i|$ and the set of

$|f_j - np_j|$ with $j \neq i$ become uncorrelated as m grows. Under this assumption, equation (1.5) can be considered a good approximation for the variance of I_m .

Substituting (1.2) and (1.3) in (1.4) and (1.5) one has

$$E(I_m) \approx \frac{1}{\sqrt{2n\pi}} \sum_{i=1}^m \sqrt{p_i(1-p_i)} \quad (1.7)$$

$$\text{Var}(I_m) \approx \frac{\pi-2}{4n\pi} \sum_{i=1}^m p_i(1-p_i) \quad (1.8)$$

In addition to m large, assuming also $p_i \ll 1$ and knowing that $\sum_{i=1}^m p_i = 1$, one can obtain simpler formulas for (1.7) and (1.8) as follows:

$$E(I_m) \approx \frac{1}{\sqrt{2n\pi}} \sum_{i=1}^m \sqrt{p_i} \quad (1.9)$$

$$\text{Var}(I_m) \approx \frac{\pi-2}{4n\pi} \quad (1.10)$$

Therefore, for $n \rightarrow \infty$ and m large, I_m has an asymptotic normal distribution whose mean and variance are approximately given by (1.7) or (1.8) or by (1.9) and (1.10).

The simpler approximate formulas (1.9) and (1.10) were used to calculate the value of K_{I_m} presented in chapter 5.

Some preliminary Monte Carlo experiments were performed in order to evaluate the accuracy and robustness of the estimates of $E(I_m)$ and $\text{Var}(I_m)$ given by equations (1.7) through (1.10). Two multinomial distributions were chosen. The probabilities p_i $i=1, \dots, m$ were defined as follows:

$$p_i = \frac{1}{m} \quad i=1, \dots, m \quad \text{for distribution 1,}$$

and

$$p_i = \frac{1}{2}, \quad p_i \approx \frac{p_{i-1}}{2} \quad i=2, \dots, m \quad \text{for distribution 2.}$$

For each distribution, two cases were considered: few ($m=10$) and many ($m=100$) possible different outcomes.

The mean and variance of I_m for each case of the above mentioned distributions were estimated by comparing the expected number of outcomes with the number of outcomes obtained from a process of random sampling using a linear congruential pseudo-random number generator. The null hypothesis (H_0) that equation (1.7) or (1.9), and (1.8) or (1.10) yield the correct values of $E(I_m)$ and $\text{Var}(I_m)$, respectively, was tested. Confidence intervals for $E(I_m)$ and $\text{Var}(I_m)$ were obtained using standard statistical procedures [Hogg78].

In order to reduce the chances of a type I error, i.e., reject H_0 when it is true, a 1% significance test was chosen. The size of the corresponding 99% confidence interval for the non-rejection of H_0 was reduced through multiple independent replications of the experiment. A number of replications equal to 1000 yielded a confidence interval which was considered reasonably narrow for our purposes.

Tables 1.1 and 1.2 present the various estimates of the mean and variance of I_m , respectively, obtained from equations (1.7) through (1.10) and those obtained from the sampling procedure described above. The following notation applies to tables 1.1 and 1.2:

- dist.* type of distribution;
- m* number of possible different outcomes;
- n* sample size;
- mf* multiplying factor for all values in this row;
- ll* lower limit of the 99% confidence interval;
- mean* sample mean of estimates;
- var.* unbiased estimator of the population variance;
- ul* upper limit of the 99% confidence interval.

From Table I.1 we can see that, as long as $n \gg m$, the value yielded by equation (1.7) seems to be a good estimate of $E(I_m)$. The formula given in equation (1.9) overestimates I_m . When $n \approx m$ both formulas overestimate I_m .

<i>dist.</i>	<i>m</i>	<i>n</i>	<i>mf</i>	1.7	1.9	99% C.I. for $E(I_m)$		
						<i>ll</i>	<i>mean</i>	<i>ul</i>
1	10	100	10^{-1}	1.1888	1.2818	1.1880	1.1905	1.2150
		1000	10^{-2}	3.7847	3.9894	3.7438	3.8175	3.8912
	100	100	10^{-1}	3.9894	3.9894	3.8454	3.6712	3.6989
		1000	10^{-1}	1.2552	1.2818	1.2412	1.2490	1.2588
2	10	100	10^{-2}	8.1188	9.3507	7.7055	7.9340	8.1825
		1000	10^{-2}	2.5887	2.9570	2.4914	2.5819	2.6324
	100	100	10^{-2}	8.3939	9.8313	7.7709	7.9993	8.2277
		1000	10^{-2}	2.8557	3.0457	2.5547	2.8259	2.8971

Table I.1

<i>dist.</i>	<i>m</i>	<i>n</i>	<i>mf</i>	1.8	1.10	99% C.I. for $Var(I_m)$		
						<i>ll</i>	<i>var.</i>	<i>ul</i>
1	10	100	10^{-4}	8.1781	9.0845	8.1119	9.0880	10.2278
		1000	10^{-5}	8.1781	9.0845	7.7782	8.8921	9.8027
	100	100	10^{-4}	8.9937	9.0845	8.9441	9.9900	11.2605
		1000	10^{-5}	8.9937	9.0845	8.2038	9.1631	10.3339
2	10	100	10^{-4}	8.0548	9.0845	7.0472	7.8712	8.8789
		1000	10^{-5}	8.0548	9.0845	8.7081	7.4925	8.4399
	100	100	10^{-4}	8.0548	9.0845	8.8388	7.8383	8.8143
		1000	10^{-5}	8.0548	9.0845	7.0382	7.8590	8.8832

Table I.2

though the estimate given by equation (1.7) is less than 10% above the sample mean for the cases considered in Table I.1.

For all cases presented in Table I.2, the sample variance, as expected, was a little bit bigger than the estimate given by equation (1.8). The formula given in equation (1.10), however, overestimates the variance in most cases presented in Table I.2. In fact, a good estimate of the variance of I_m seems to be the arithmetic mean of the values given by equations (1.8) and (1.10). At least, this value would not cause H_0 to be rejected in any of the cases presented in Table I.2. Nevertheless, further investigation is required.

Appendix II

Table II.1 shows the stack distance probabilities used in the three phase-transition models for the generation of the original traces. The generation of unimodal, bimodal and trimodal working-set density functions was accomplished by using one, two and three localities, respectively. For the

stack dist.	unimodal	bimodal		trimodal		
	loc. 1	loc. 1	loc. 2	loc. 1	loc. 2	loc. 3
1	0.02	0.1	0.02	0.3	0.02	0.02
2	0.02	0.1	0.02	0.3	0.02	0.02
3	0.02	0.1	0.02	0.2	0.02	0.02
4	0.02	0.1	0.02	0.1	0.02	0.02
5	0.02	0.2	0.02	0.05	0.02	0.02
6	0.05	0.2	0.02	0.01	0.02	0.02
7	0.05	0.05	0.02	0.01	0.02	0.02
8	0.05	0.05	0.02	0.01	0.02	0.02
9	0.1	0.02	0.02	0.01	0.02	0.02
10	0.1	0.02	0.02	0.01	0.02	0.02
11	0.1	0.02	0.02		0.1	0.05
12	0.1	0.01	0.02		0.1	0.05
13	0.05	0.01	0.02		0.1	0.05
14	0.05	0.01	0.02		0.1	0.05
15	0.05	0.01	0.02		0.1	0.05
16	0.02		0.01		0.1	0.05
17	0.02		0.01		0.05	0.05
18	0.02		0.01		0.05	0.05
19	0.02		0.01		0.05	0.05
20	0.02		0.01		0.05	0.05
21	0.01		0.02			0.05
22	0.01		0.02			0.05
23	0.01		0.02			0.05
24	0.01		0.02			0.03
25	0.01		0.02			0.02
26	0.01		0.02			0.02
27	0.01		0.02			0.02
28	0.01		0.02			0.02
29	0.01		0.02			0.02
30	0.01		0.02			0.02

Table II.1

unimodal, a locality composed of pages 1-30 was defined. For the bimodal, locality 1 is composed of pages 1-15, and locality 2 of pages 1-30. For the trimodal, locality 1 is composed of pages 1-10, locality 2 of pages 11-30, and locality 3 of pages 1-30.

A transition between localities was taken after k references within one specific locality. The variable k had an exponential distribution with mean equal to 50 references. In the case of the trimodal, when leaving a specific locality, the transitions to either of the remaining ones were made to be equally likely.