EXPERIMENTAL DESIGN OF A GENERATIVE MODEL BASED ON WORKING SET SIZE CHARACTERIZATIONS

by

T-Y. P. Lee

Memorandum No. UCB/ERL M82/56

19 July 1982

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Experimental Design of a Generative Model Based on Working Set Size Characterizations*

Tzong-yu Paul Lee

Comuter Science Divison
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

*The research reported here has been supported by the National Science Foundation under grant MCS80-12900.

ACKNOWLEDGMENT

The author wishes to express his gratitude for the encouragement and support of his research advisor, Professor Domenico Ferrari, who introduced him to the topic discussed in this report. The author also benefited very much from discussions with members of the PROGRES research group, in particular, Juan Porcar.

CHAPTER 1

Introduction

In the study of program behavior, memory referencing patterns have long been a subject of interest. Various memory management policies were proposed and implemented to optimize the utilization of system resources and increase system throughput. On the other hand, techniques were employed to enable compilers to generate better-behaved code in order to achieve the same purpose. Various methods for restructuring programs so as to improve their referencing behavior have also been investigated [Ferr76a].

Very often, in these studies, program reference strings were used to compare the performances of various memory management policies or to validate models of program behavior. These strings were often collected from real programs running on existing systems. The process of gathering reference strings is rather tedious and generally very expensive. To be precise, most strings are gathered by interpretive execution. There are clear advantages in using artificial strings if they reflect the behavior of the real strings. This leads to the study of generative models. The goal of these models is to use relatively few parameters and relatively fast algorithms to generate artificial reference strings which can be employed in various trace-driven simulation experiments.

It is usually not necessary to reproduce a program's behavior exactly; consequently, the generative model will try to reproduce those properties of a string which are deemed to be important in the context in which the artificial string is to be used. In this study, memory consumption and page fault rate are chosen as the important aspects to be reproduced. Our primary context is that of the working set policy [Denn68a], but we shall also be comparing a real and several artificial strings under the page fault frequency policy [Chu76a] and the least recently used policy [Matt70a]. For convenience, these policies shall be referred to as WS, PFF, and LRU respectively.

Artificial strings will be evaluated in three steps. First, a comparison of first moment results with those from the real string. If this is satisfactory, the distributions of some of the indices will be compared. The last step would be to investigate the dynamic behavior of the reference string, which is essentially a finite time series, with statistical techniques [Bril75a]. Most of these techniques are independent of the memory management policies chosen.

The generative model evaluated here, which is based on a working set characterization, was proposed by Ferrari [Ferr81a] and first implemented by Dutt [Dutt81a]. The theoretical aspects and the design of the experiments is detailed in the next two chapters. The results and their analysis are presented in Chapter 4. Finally, in the last chapter, conclusions are drawn from the outcome of the experiment and directions for further research are provided.

CHAPTER 2

Theoretical Aspects

2.1. Background

The goal of a generative model is to provide a sequence of page references whose characteristics of interest are similar to those of the modeled program. The generative model evaluated here is based on the sequence of working set size(s) and possibly on other parameters. While it is desirable to reproduce the dynamic behavior of the modeled program as accurately as possible, it is also important to minimize the number of parameters needed for the generative model. A sequence of working set sizes is generally redundant in its information content. It can be completely specified as a sequence of the pairs (t_i, w_i), where t_i is the time at which the working set size (wss) curve changes its slope, and w_i is the wss at that time. This sequence is the input to the generative model used in the experiment.

When a sequence of wss's is given, a window size T must always be specified. It is clear that the string generated by the model is by no means the same as the one of the modeled program. However, it is hoped that, with appropriate generation algorithms, the generated string will possess the essential characteristics of the modeled one. An example is given here to illustrate some fundamental notions. Assuming that the length of a string of page references is 5, the window size is 3, and the page set is {a,b}, Table 2.1 contains all possible sequences, their wss sequences, and the relationships among them.

There are 2^5 possible strings; however, there are only 16 distinct wss sequences. Strings with the same wss sequence have the same wss characterization, and are members of the same class. As far as the wss characterization is concerned, any of the strings in that class would represent well the others. There exist even stronger equivalence classes, i.e., those of strings equivalent under renaming. Two sequences are equivalent under renaming if there exists a one-to-one mapping of the pages used in one sequence to those used in the other sequence. Two strings which are equivalent under renaming are also wss equivalent, i.e., have the same wss characterization; the converse, however, is not necessarily true. One simple example is as follows.

String 1 : a b c a b c a b c ...
String 2 : b c a b c a b c a ...

The mapping f between two page sets is f(a) = b, f(b) = c, and f(c) = a.

In practice, the assignment of page numbers to program pages is sequential. Also, the way in which programs are laid out in their virtual address space usually reflects some degree of sequential referencing pattern. These facts have no impact on working set characterizations. Note that, however, any page prefetching strategy could very well use the knowledge of sequentiality implied by page numbers.

Table 2.1

Examples of Reference Strings		
string	wss	string under renaming
a a a a a	1 1 1 1 1	b b b b b
a a a a b	1 1 1 1 2	b b b b a
a a a b a	1 1 1 2 2	b b b a b
a a a b b	1 1 1 2 1	b b b a a
a a b a a	1 1 2 2 1	b b a b b
a a b a b	1 1 2 2 2	b b a b a
a a b b a	1 1 2 1 2	b b a a b
a a b b b	1 1 2 1 1	b b a a a
a b a a a	1 2 2 1 1	b a b b b
a b a a b	1 2 2 1 2	b a b b a
a b a b a	1 2 2 2 2	b a b a b
a b a b b	1 2 2 2 1	b a b a a
a b b a a	1 2 1 2 1	b a a b b
a b b a b	1 2 1 2 2	b a a b a
a b b b a	1 2 1 1 2	b a a a b
a b b b b	1 2 1 1 1	b a a a a
b a a a a	1 2 1 1 1	a b b b b
b a a a b	1 2 1 1 2	a b b b a
b a a b a	1 2 1 2 2	a b b a b
b a a b b	1 2 1 2 1	a b b a a
b a b a a	1 2 2 2 1	a b a b b
b a b a b	1 2 2 2 2	a b a b a
b a b b a	1 2 2 1 2	a b a a b
b a b b b	1 2 2 1 1	a b a a a
b b a a a	1 1 2 1 1	a a b b b
b b a a b	1 1 2 1 2	a a b b a
b b a b a	1 1 2 2 2	a a b a b
b b a b b	1 1 2 2 1	a a b a a
b b b a a	1 1 1 2 1	a a a b b
b b b a b	1 1 1 2 2	a a a b a
b b b b a	1 1 1 1 2	a a a a b
b b b b b	1 1 1 1 1	a a a a a

2.2. Feasibility of single T generation

By 'single T generation' we refer to the generation of an artificial string based on one wss sequence, corresponding to one value of T. It is important to characterize the properties of the so-called wss sequences. In principle, all possible reference strings form a vector space S^n where S is the page set and n is the length of the string. By specifying a particular sequence of wss's, we have focused on a subset of this vector space in which every element has the same wss characterization. Essentially, the input to the generative model is the given wss sequence, which is associated with some window size T. It should be clear that the string generated by the model is generally not unique; therefore, the wss sequence associated to the artificial string with a different window size will be different from that of the modeled program under the same conditions. Performance indices of other kinds are usually different also. The motivation for introducing appropriate strategies into the generation algorithm is to obtain that the artificial string represent the real string closely not only for the wss dynamics but also for other indices of interest.

The next issue to be investigated is clearly that of the properties of a wss sequence [Ferr81a] [Ferr81b]. Before we introduce the necessary and sufficient conditions for a sequence of integer numbers to be usable to generate an artificial string, some definitions are needed.

Definition 2.1

Given an integer bound T, a string S = $s_1 \cdots s_i \cdots s_n$ is called a bounded positive continuous string if the following three conditions hold:

(i) $s_1=1$
(ii) $0<s_i\leq T$ for $1\leq i\leq n$
(iii) $|s_i-s_{i-1}| \leq 1$ for $2\leq i\leq n$

Definition 2.2

A string S is a wss (working set size) string with integer bound parameter T if there exists a page reference string R = $r_1 \cdots r_n$ such that, with window size T, the wss string of R coincides with S.

Definition 2.3

Given a wss string S = $s_1 \cdots s_i \cdots s_n$, the decrement count at time t, d_t, is the number of decrements in substring $s_t s_{t+1} \cdots s_{t+T-1}$.

With the above definitions, we state the following theorem which gives the condition for the existence of at least one artificial string corresponding to a given string of integers.

Theorem 2.1

A bounded positive continuous string is a wss string with parameter T if and only if $d_t<s_t$ for all t's.

The proof of the theorem is constructive, and is not given here. It can be found in [Ferr81a].

2.3. Feasibility of double T generation

By 'double T generation' we refer to the generation of an artificial string based on two wss sequences. By specifying two wss sequences with two different window sizes, the subset of legitimate strings is further constrained. With the use of appropriate strategies, it is expected that the artificial string generated will more closely resemble the modeled program's behavior than that generated by a single T wss characterization. Again, it is essential to investigate the feasibility of such generation. The main result is proved in [Ferr82a] and is stated here without proof.

The following definitions are needed to introduce the theorem.

Definition 2.4

A working set size (wss) string S_1 with parameter T_1 is said to be greater than or equal to a working set size string S_2 with parameter T_2 if $T_1 > T_2$ and $s_{1_t} \geq s_{2_t}$ for all t. We indicate this by the notation $S_1 \geq S_2$.

The properties described below are based on two wss strings with parameters T_1 and T_2 where $T_1 > T_2$. Strings S_1 and S_2 are represented as s_{1_1} ... s_{1_t} ... and s_{2_1} ... s_{2_t} ... respectively. These two strings are assumed to have the same lengths. Care should be taken to ensure that edge problems will not arise.

Definition 2.5

1. Property (i) holds if $s_{1_t} = s_{1_{t-1}} + 1$ implies $s_{2_t} = s_{2_{t-1}} + 1$ for all t
2. Property (ii) holds if $s_{1_t} = s_{1_{t-1}} - 1$ implies $s_{2_{t-T_1+T_2}} = s_{2_{t-T_1+T_2-1}} - 1$ for all t
3. Property (iii) holds if $s_{2_t} = s_{2_{t-1}} - 1$ and $s_{1_{t+T_1-T_2}} \geq s_{1_{t+T_1-T_2-1}}$ implies the existence of a unique $k \in (t, t+T_1-T_2]$, such that $s_{2_k} = s_{2_{k-1}} + 1$ and $s_{1_k} \leq s_{1_{k-1}}$ for all t

Theorem 2.2

A reference string R that has the wss characterizations represented by S_1 and S_2 with window sizes T_1 and T_2 exists if and only if $S_1 \geq S_2$ and properties (i)-(iii) hold.

2.4. The problems of flat-faults

An obvious way to get a wss characterization for the generative model is to get it from a real trace. Caution needs to be taken, however, in this process. A definition will be introduced to illustrate the problem.

Definition 2.6

A flat-fault occurs at time t if the working set size at t is the same as working set size at t-1 and there is a page fault at time t.

Basically, a flat-fault occurs when there is a new page coming into the working set and an old page dropping out of working set at the same time. This phenomenon is very unlikely to occur with any practical value of T, and is discussed in [Lee82a]. The proof of Theorem 2.1 does not depend on the assumption that the given wss characterization has no flat-faults. This is confirmed in a different way by Theorem 2.3 below. However, the proof of Theorem 2.2 depends on the assumption that there are no flat-faults in the two given wss characterizations. A natural question is whether similar theorems for the existence of an artificial string hold when S_1 and/or S_2 contain flat-faults. If no such general results exist, it is necessary for the generation program to detect such situations, or the wss characterization(s) should be checked for flat-faults before they are used in the generation program. These issues are discussed in [Lee82a], and the results are summarized here without proof.

2.4.1. single T generation

An example of flat-fault is given in Table 2.2. With window size equal to 4, a flat-fault occurs at time 5, when page 'a' drops out of the working set and page 'd' enters the working set. In the same vein, a flat-fault occurs at time 7 with window size equal to 3. If the given wss characterization of the real trace contains flat-faults with respect to window size T, the existence of an artificial string is guaranteed by the following theorem. The proof of this theorem can be found in [Lee82a].

Theorem 2.3

There exists a reference string R corresponding to a given wss sequence S extracted from a real trace which contains flat-faults.

2.4.2. double T generation

It is unfortunate that no existence theorem similar to Theorem 2.3 can be proved for double T generation if either of the two wss characterizations contains flat-faults. Examples are given in [Lee82a] to show that, without knowing when flat-faults occur, no artificial string can be generated from wss characterizations containing flat-faults. More information is needed for the model to generate an artificial string in this case.

2.5. The deadline of a page reference

The proofs of Theorems 2.1 and 2.2 are constructive. In the generation algorithms on which the proofs are based, those pages that need to be referenced again in order to stay in the working set are referenced in a FIFO manner. The time frame in which a page has to be re-referenced in order to remain in the working set is referred to as the page's 'deadline' here. Referencing in a FIFO manner is the most conservative way to satisfy the requirement and facilitates the proofs. However, pages are generally not referenced in a cyclic manner by real programs. It will be desirable to re-reference pages in other ways, but this cannot be comfortably done unless we know that all the deadlines can still be met each time a decision is made to re-reference an old page in a non-FIFO manner.

The deadlines of pages in the working set are difficult to describe in a simple and compact form. Examples are given instead in Tables 2.3, 2.4 and 2.5 to illustrate the notion of deadlines and interdependencies among the deadlines of different pages.

In the first example, a page 'x' is referenced at time 10 and there is an increase in the wss from time 14 to time 15. In order to allow such an increase, page 'x' has to be re-referenced before (but not at) time 15. This is the deadline for page 'x' at time 10. The asterisk at time 16 corresponds to the first forbidden location for page 'x'.

Table 2.2

Flat-fault Example							
time	1	2	3	4	5	6	7
page	a	b	c	b	d	d	c
wss (T=4)	1	2	3	3	3	3	3
flat-fault	.	.	.	.	*	.	.
wss (T=3)	1	2	3	2	3	2	2
flat-fault	.	.	.	.	.	.	*

Table 2.3

Deadline Example 1 (T = 5)											
time	.	.	.	10	11	12	13	14	15	16	.	.
wss	.	.	.	4	4	4	4	4	5	.	.	.
page	.	.	.	x	.	.	.	.	*	.	.	.

Table 2.4

Deadline Example 2 (T = 5)											
time	.	.	.	10	11	12	13	14	15	16	.	.
wss	.	.	.	4	4	4	4	4	4	.	.	.
page	.	.	.	x	.	.	.	.	.	*	.	.

A similar example is given in Table 2.4 to illustrate a different situation. The difference is that here the string has the same wss at times 14 and 15. Under the assumption that there are no flat-faults, page 'x' has to be re-referenced before or at time 15.

In general, when page references are generated by the model, the deadlines have to be continuously updated depending on the page reference just generated and on the knowledge of wss variations in a forward window of size T.

In the third example (Table 2.5), page 'a' is referenced at time 1 and an increase in wss is to take place from time 8 to time 9. A simple deadline for page 'a' is before (but not at) time 9. However, this deadline is unrealistic, since at time 8 we will have to select a new page for the working set. The actual deadline should be moved to time 8 (before but not at). The page referenced at time 2 is 'b', and there is an increase in wss also from time 9 to time 10. Thus, the deadline for 'b' is time 10. However, it is clear that time slots 8 and 9 are both reserved for new pages. A better deadline will be time 8. But there is one more factor that affects the actual deadlines for both pages 'a' and 'b' now. It is the interaction between the two requirements. There is only one page that can be referenced at time 7. If both pages 'a' and 'b' wait until this time, these two deadlines cannot be met simultaneously. Therefore, realistic sets of deadlines at time 2 are as follows :

(1) page 'a' at 8 and page 'b' at 7

or

(2) page 'a' at 7 and page 'b' at 8

If either of the above requirements is satisfied, the deadlines are met.

Table 2.5

Deadline example 3 (T = 8)													
time	1	2	3	4	5	6	7	8	9	10	11	12	.	.
wss	1	2	3	3	3	3	3	4	5	6	7	.	.	.
page	a	b	c	.	.	.	.	d	e	f	g	.	.	.

It is now easy to extend this observation to time 3. A realistic set of deadlines for pages 'a', 'b', and 'c' would be any of the permutations of 6, 7, and 8. Furthermore, after a page for time 4 is selected from {a,b,c}, the deadlines have to be updated. A realistic deadline policy would be to keep enough page slots free in the forward window of size T for the pages in the working set. However, there seems to be no simple way to maintain this information in a compact form. Choosing an efficient data structure and an efficient algorithm appears quite difficult.

We shall not try to determine a general method for dealing with the deadlines for the pages in the working set. The problem is only of theoretical interest, since practically the window size T is much greater than the total number of pages in use. In summary, it is sufficient to approximate the deadlines for the pages in the working set. The problem is certainly even more complex when two sets of wss characterizations are given, but it can be dealt with by a similar approach. In particular, we need to concentrate on the wss characterization corresponding to the smaller T since this characterization has a more stringent set of constraints.

CHAPTER 3

Design of the Experiments

3.1. Implementation Issues

There are several implementation-related issues needing resolution, which are discussed in this section.

3.1.1. Boundary conditions

Since the artificial strings generated are of finite length, care should be taken in the algorithms to ensure that the generation process will work correctly also at the string boundary.

3.1.1.1. starting conditions

The string generation algorithm (and the program we implemented) always assumes that the initial working set of the program is empty. The only concern that could arise is the effect of this assumption on the performance indices. If the string is long enough in comparison with the time to reach the mean working set size, the contribution of the starting conditions to the long-term averages can be neglected. Indeed, this is the case for the experiments carried out in this study.

3.1.1.2. ending conditions

The string generation algorithm looks ahead T (or T_1) units of time. Depending on the future changes of the working set size, the currently referenced page will be put into the appropriate set. If our input wss string has the same length as the reference string to be generated from it, the algorithm will stop T time units before the end, i.e., before its task is completed. Something must be done to resolve this difficulty.

For single T generation, the algorithm could assume that the future unspecified working set sizes are the same as the last size known to the program [Dutt81a]. This extension is feasible because of the following theorem. The proof of this theorem is in Appendix A.

Theorem 3.1
If $S = s_1 \cdots s_n$ is a wss string obeying the conditions of Theorem 2.1 for window size T, then the extended string $s_1 \cdots s_n s_n \cdots s_n$ of length $n+T$ satisfies the conditions of Theorem 2.1.

The wss characterizations for double T generation can also be extended. The proof of Theorem 3.2 is somewhat more complicated than that of Theorem 3.1 and can also be found in Appendix A[1]. Note that both wss characterizations are assumed to have the same length.

[1]There is a limit on the length of the extension. Theorem 3.2 should be interpreted carefully. See Appendix A.

Theorem 3.2

Given two wss strings $S_1=s_{1_1} \cdots s_{1_n}$ and $S_2=s_{2_1} \cdots s_{2_n}$ obeying the conditions of Theorem 2.2 with window sizes T_1 and T_2 respectively, then there exists a a reference string R of length n having wss characterizations S_1 and S_2 with window sizes T_1 and T_2 respectively which is generated by the extended wss strings $s_{1_1} \cdots s_{1_n} s_{1_n} \cdots s_{1_n}$ of length $n+T_1$ and $s_{2_1} \cdots s_{2_n} s_{2_n} \cdots s_{2_n}$ of length $n+T_1$.

However, the approach taken in our experiments is to extract longer wss characterizations from a real trace, so that reference strings of the desired length can be generated without extensions.

3.1.2. Meeting the deadlines

While there are usually many choices for the next reference, the re-referencing deadlines for all pages in the working set should be met in order for the wss characterization(s) to be faithfully reproduced. It has been shown in the previous section how the deadlines for those pages can be defined; however, it would be complicated to update all deadlines after each new reference is generated. Even though in real traces the expected number of consecutive references to the same page is small, it is not safe to use the original deadline for each individual page. Better alternatives to this approach are implemented in the generation algorithms, which will now be described.

3.1.2.1. single T generation

An approximation to the exact solution of the deadline problem is to use the original deadline for each page, which is obtained when that page is referenced and put into the candidate queue, i.e., the queue of the pages to be re-referenced before their deadlines. Before a page is selected from the candidate queue in a non-FIFO manner, the deadline of the first page is checked against the current time plus the maximum possible working set size. If the deadline falls short of the value of this sum, the first page is referenced immediately to prevent potential problems in the future. Since there are enough reserved page slots for every possible page in the current working set and for the possible new pages which will join the working set at the times of future wss increases, the deadlines will be all met. A simpler approximation is used in the current implementation. Instead of using the maximum possible working set size, the current working set size is used in deadline checking. The generation program has built-in checks for deadline violations.

3.1.2.2. double T generation

A similar argument can be made for an approximation for the double T case. Instead of using the maximum possible working set size with the larger T, the current working set size with the larger T is used for deadline checking. The program has also in this case built-in checks for deadline violations.

3.1.3. Parameters of the model

The parameters of the model are estimated from a real trace. This trace was obtained from the interpretive execution of an APL program on an IBM 360/91 machine. Except for the wss characterizations, which were obtained from the first 550,000 references, all parameters were derived for the first 500,000 references. Various performance indices of the real trace, later used for comparisons, were also computed by trace-driven simulations from these 500,000 references.

Three wss characterizations were obtained with window sizes of 5000, 10000, and 20000 references. For single T generation, the wss characterization with window size 10000 was used. For double T generation, the other two were used. A characterization is given in the form of a sequence of (t,w) pairs, where t is the time at which the wss changes and w is the value of the wss at that time. The numbers of pairs are 1157, 619, and 525 for window sizes of 5000, 10000, and 20000, respectively. The nominal window size of 10000 was chosen for two reasons. First, this window is not so short as to obliterate the program's phase transition behavior [Dutt81a] and not so long as to require too much memory space. Secondly, in the neighborhood of window size 10000, the space time product shows rather stable values.

The coefficient of resilience is defined here as the probability that the page referenced next is the same as the currently referenced page. In essence, this is the probability of referencing the top of the stack in an LRU environment (it is often called d_1 in the context of the stack distance distribution [Spir77a]). The estimate of this parameter from the real trace yielded the value 0.544. When this number is large enough, a reference string can be stored as a sequence of $(page, count)$ pairs in order to minimize storage requirements. This has been done in this experiment.

The number of distinct pages in the first 500,000 references in the real trace was found to be 110. The relative referencing frequencies of the 110 referenced pages were measured, sorted and plotted. This frequency distribution can be found in Appendix B. It is interesting to observe that the most popular page accounts for 25 percent of the references. Also, 20 percent of the pages account for 86 percent of the references.

3.1.4. Performance indices

Various performance indices were chosen for comparison between real and artificial strings. To compute the space time product, the page wait time has been assumed to be constant and equal to 10000 references. The primary performance indices considered in the various contexts are listed below.

3.1.4.1. WS environment

Mean working set size, page fault rate, space time product, working set size distribution, and interfault time distribution are the primary indices we are interested in when the WS policy is used.

3.1.4.2. PFF environment

Mean working set size, page fault rate, space time product, working set size distribution, and interfault time distribution are the primary indices of concern in the PFF case. The parameter of the PFF algorithm, i.e., the threshold of interfault times, was chosen to equal 1543 time units. This is the value of the mean interfault time obtained under the WS policy with window size 10000.

3.1.4.3. LRU environment

Page fault rate, space time product, interfault time distribution, and stack distance distribution are the primary indices of concern in the LRU case. For the stack distance distribution, the probability d_1 of referencing the top of the stack is particularly important. The parameter of the LRU algorithm, i.e., the fixed partition size, was chosen to be 21 page frames. This is the mean working set size with a window size of 10000. According to [Denn72a], this choice for LRU should produce the same page fault rate as the WS policy with window size 10000.

3.2. Overview of the Experiments

3.2.1. Artificial Strings

Twelve artificial strings of length 500,000 each were generated. Each string was evaluated and compared with the real string. These twelve strings are named T00, T01, T02, T10, T11, T12, TT00, TT01, TT02, TT10, TT11 and TT12. These names reflect the three control variables of the experiment : the number of wss characterizations, the ways to select new pages when needed, and the ways to reuse pages already in the working set.

In the case of single T generation, the string's name begins with T. In the case of double T generation, the string's name begins with TT. The corresponding artificial strings are called T** and TT** respectively. The first digit following T or TT corresponds to the way old pages are reused. It is 0 if the old pages are reused in a FIFO manner; this means that pages are selected from the candidate queue in the order in which they were put in. It is 1 if the previously referenced page is reused with a given probability (coefficient of resilience) , and otherwise in a FIFO manner; it is not too difficult to verify that the number of consecutive references to the same page is distributed geometrically. The second digit following T or TT corresponds to the way a new page is selected. It is 0 if the new page is selected from the external (new page) queue in a LIFO manner; essentially, this scheme recycles the pages put back to the external queue whenever possible. It is 1 if the new page is selected from the external queue in a circular manner; the length of the external page queue is initially set to the total number of distinct pages. The digit is 2 if a new page is selected from the external queue according to a pre-specified program profile. The usage record of each page is continuously updated so that the appropriate page can be chosen to match the intended program profile.

3.2.2. Data structure

3.2.2.1. single T generation

There are basically three singly-linked lists (queues) : the candidate queue (C), the forbidden queue (F), and the external queue (E). These queues correspond the those used in the proof of Theorem 2.1 [Ferr81a]. A page exists in one and only one queue at any given time. Intuitively, the pages in the candidate queue are those which can be referenced without increasing the size of the working set, and which are to be re-referenced by a deadline in order to remain in the working set. The pages in the forbidden queue are those pages which cannot be re-referenced until they drop out of the working set. When a forbidden page drops out of the working set, it is put back into the external queue. All pages are initially in the external queue. When the working set size increases, a new page is selected from this queue according to one of the specified strategies. Pages are sent back to this queue only from the forbidden queue.

3.2.2.2. double T generation

In principle, we can operate with two sets of queues, each corresponding to one wss characterization : that is, one set will consist of C_1, F_1, and E_1, and another set of C_2, F_2, and E_2. Each page is at any given time in one (and only one) of the three queues in each set. However, if these six queues are implemented as described, it will be necessary to calculate a large number of intersections of two queues, one from each set. This is undesirable from the viewpoint of the speed of the generation algorithm. Therefore, a data structure consisting of five doubly-linked lists (queues) was implemented. Each queue in the structure corresponds to a particular intersection of the two queues mentioned above. Notice that, fortunately, not all possible intersections of queues are needed in the generation of reference strings. The five queues required are C_1C_2, F_1F_2, E_1E_2, C_1E_2, and C_1F_2.

All pages are initially in the E_1E_2 queue. When both wss characterizations contain a wss increase, a new page is selected from the E_1E_2 queue according to one of the specified algorithms. When both wss characterizations contain a non-increasing transition, a page is selected from the C_1C_2 queue. When the wss with the larger T does not require to be increased and the other needs to be increased, a page is selected from the C_1E_2 queue.

When a page is chosen, the queue to which this page should be added is to be selected. This page is in two working sets of different sizes. When both working sets require this page be re-referenced by a deadline in order to remain in both working sets, the page is put into the C_1C_2 queue. When this page has to drop out of both working sets later on, the page is put into the F_1F_2 queue. When this page has to remain in the working set with the larger T, but has to drop out of the working set with the smaller T, the page is put into the C_1F_2 queue. Notice that, by Theorem 2.2, all other combinations of transitions cannot occur, either due to Properties (i)-(iii) or due to the no flat-faults assumption.

A page stays in the F_1F_2 queue until it drops out of both working sets, at which time the page is released and moved to the E_1E_2 queue. Similarly, a page stays in the C_1F_2 queue until it drops out of the working set with the smaller T, at which time the page is released and sent to the C_1E_2 queue. No other transitions of pages between queues are possible. It should be noted that such arrangement of the data structure also maintains the chronological ordering of the pages in each queue. Therefore, no search is needed to select and delete a page at each generation of a page reference when the FIFO strategy is used.

CHAPTER 4

Experimental Results and Their Analysis

As mentioned in Chapter 3, twelve artificial strings were generated and named T00, T01, T02, T10, T11, T12, TT00, TT01, TT02, TT10, TT11, and TT12. The generation of 500,000 references with a single window size (hence, a single wss characterization) takes from 275 seconds to 421 seconds of VAX-11/780 CPU time depending on the options chosen. The generation of 500,000 references with two window sizes takes from 306 seconds to 466 seconds of VAX-11/780 CPU time depending on the options chosen. The surprisingly efficient double T generation comes from the carefully planned structure of the queues, which eliminates the need to do linear searches on them in most cases.

The strings were run in a trace-driven simulation context under various memory management policies, and their performance indices were compared with those produced under the same policies by the real string. These comparisons are discussed in the following sections. Notice that string T00 is essentially the same as the string generated in an earlier experiment [Dutt81a].

4.1. Characterization of Artificial Strings

The program profiles of all twelve artificial strings were obtained. The program profile for the real string is shown in Appendix B. Two statistics are listed in Table 4.1 for comparisons : the total number of distinct pages used in each string, and the coefficient of resilience. The number of distinct pages used in T02, T12, TT02, and TT12 would reach 110 if the string generated were infinitely long. However, due to the very small probability densities at the tail of the distribution , only a fraction of all pages are actually used in generating 500,000 references.

4.2. WS policy

Artificial strings were executed under the WS policy with window size 10000 if they were generated with one T, and with window sizes 20000 and 5000 if they were generated with two T's. Their performance indices were found to be exactly the same as those of the real string executed under the same window size(s). This was expected, but strengthened our confidence in the correctness of the generation programs. The results under the WS policy with a few different window sizes for the real string are given in Table 4.2. The working set size distribution for a window size of 10000 is reported in Appendix C.

When generating artificial strings with two window sizes, it is of interest to investigate the accuracy of the characterization between the two Ts used in the generation phase. For a few intermediate window sizes, the results are summarized in Tables 4.3, and 4.4, and 4.5. To compare the distributions of the working set size and the interfault time with window size 10000, refer to Appendices C and G. Not only the first moment results are very close (within

Table 4.1

program profile		
string	number of distinct pages	coefficient of resilience
T00	56	0.000
T01	110	0.000
T02	80	0.000
T10	56	0.544
T11	110	0.544
T12	80	0.544
Real	110	0.544
TT00	78	0.000
TT01	110	0.000
TT02	80	0.000
TT10	78	0.544
TT11	110	0.544
TT12	80	0.544

Table 4.2

WS results for the real string						
window size	mean wss	max wss	changes of slope	space time product	page fault rate	max interfault time
20000	26.17	78	525	1.10E8	0.000562	111695
15000	23.67	67	554	1.07E8	0.000592	111695
10000	20.90	56	619	1.06E8	0.000648	111695
7500	19.29	51	742	1.14E8	0.000770	65292
5000	16.90	45	1157	1.29E8	0.00118	32092

5 percent) to those of the real string, but also the distributions.

Table 4.3

WS characterization (T=7500)						
string	mean wss	max wss	changes of slope	space time product	page fault rate	max interfault time
TT**	19.38	51	772	1.18E8	0.000800	80577
Real	19.29	51	742	1.14E8	0.000770	65292

Table 4.4

WS characterization (T=10000)						
string	mean wss	max wss	changes of slope	space time product	page fault rate	max interfault time
TT**	21.05	56	613	1.05E8	0.000642	111695
Real	20.90	56	619	1.06E8	0.000648	111695

Table 4.5

WS characterization (T=15000)						
string	mean wss	max wss	changes of slope	space time product	page fault rate	max interfault time
TT**	23.71	67	532	1.05E8	0.000570	111695
Real	23.67	67	554	1.07E8	0.000592	111695

An intriguing result obtained in the double T case is summarized in the following theorem. The theorem is proved in Appendix A together with the three lemmas its proof rests upon. The basic idea is that, under wss-preserving transformations, the six strings TT00, TT01, TT02, TT10, TT11, and TT12 can be shown to be equivalent under the WS policy for all window sizes T between the two window sizes used in the generation phase.

Theorem 4.1
Strings TT00, TT01, TT02, TT10, TT11, TT12 generated from two wss characterizations with window sizes T_s and T_l, with $T_s < T_l$, have the same wss characterizations for any T such that $T_s < T < T_l$.

4.3. PFF policy

It is not too surprising to find that the accuracy of string T00 is quite low under the PFF policy. The difficulty arises since, after the artificial string manages to bring all its pages into memory, no further page faults can occur. However, if we increase the size of the page population at the time of selecting a new page in the string generation algorithm, we can quite adequately reproduce the performance of the real string under the PFF policy. Using double T generation produces a very good accuracy also. This encouraging result is due in part to the similarity between the WS and PFF policies in their dynamic and adaptive allocation of memory to processes. The results are summarized in Table 4.6. More detailed results can be found in Appendices E and I.

4.4. LRU policy

The performance of all artificial strings under LRU differs significantly from that of the real string. Even making various modifications to the original simple generation algorithm, the comparison still fails to come close. In the simulation, also the LRU stack distance distribution is obtained. This distribution for the real string is given in Appendix D. The probability d_1 of referencing the top of the stack is reported in Table 4.7. One reason for the inaccuracy is that the value of the parameter m, the number of page frames allocated, used in the LRU environment is not appropriate. The page fault rate produced with a memory allocation of 21 page frames was more than twice that produced by the WS policy with window size 10000. It is clear that some of the assumptions in Denning's paper [Denn72a] are violated. The stationarity of the reference string is probably a most unrealistic assumption in this experiment. Similar findings were also reported in the literature [Smit76a].

A new value of parameter m was obtained by trying to match the measured page fault rate of the real string under the WS policy with window size

Table 4.6

PFF characterization (I=1543)						
string	mean wss	max wss	changes of slope	space time product	page fault rate	max interfault time
T00	55.50	56	56	4.30E7	0.000112	490024
T01	20.97	67	318	1.02E8	0.000648	111695
T02	26.16	64	260	8.83E7	0.000528	111695
T10	55.5	56	56	4.32E7	0.000112	490024
T11	20.97	67	318	1.02E8	0.000648	111695
T12	27.68	64	255	8.89E7	0.000516	111695
Real	20.71	67	417	1.17E8	0.000842	80339
TT00	21.91	67	381	1.14E8	0.000776	101010
TT01	20.63	67	413	1.18E8	0.000848	80399
TT02	20.93	67	409	1.18E8	0.000836	80399
TT10	21.91	67	381	1.14E8	0.000776	101010
TT11	20.63	67	413	1.18E8	0.000848	80399
TT12	20.63	67	412	1.18E8	0.000846	80399

Table 4.7

LRU Characterization (m=21)				
string	page fault rate	max interfault time	space time product	d_1 (percent)
T00	0.217	111834	2.28E10	0.0
T01	0.217	111695	2.28E10	0.0
T02	0.217	111695	2.28E10	0.0
T10	0.100	111834	1.05E10	54.4
T11	0.100	111695	1.05E10	54.4
T12	0.099	111695	1.04E10	54.5
Real	0.00146	111416	1.63E08	54.4
TT00	0.130	111702	1.36E10	0.0
TT01	0.130	111563	1.36E10	0.0
TT02	0.130	111563	1.36E10	0.0
TT10	0.0605	111702	6.36E09	54.4
TT11	0.0606	111563	6.37E09	54.4
TT12	0.0601	111563	6.32E09	54.4

10,000 references. This new value turned out to be 31. The performance indices obtained from the LRU policy with the new m are given in Table 4.8. There is almost a one order of magnitude improvement in the accuracy of the artificial string with parameter m equal to 31 as shown in Table 4.9. However, the accuracy is still unsatisfactory. These artificial strings cannot be used in LRU-related experiments reliably.

Table 4.8

LRU characterization (m=31)				
string	page fault rate	max interfault time	space time product	d_1 (percent)
T00	0.00990	175482	1.55E9	0.0
T01	0.0102	111695	1.59E9	0.0
T02	0.0101	111698	1.58E9	0.0
T10	0.00476	175482	7.54E8	54.4
T11	0.00502	111695	7.94E8	54.4
T12	0.00504	111765	7.96E8	54.5
Real	0.000670	175392	1.19E8	54.4
TT00	0.00896	175392	1.40E9	0.0
TT01	0.00913	111695	1.43E9	0.0
TT02	0.00908	111695	1.42E9	0.0
TT10	0.00439	175392	6.96E8	54.4
TT11	0.00456	111695	7.23E8	54.4
TT12	0.00454	111695	7.20E8	54.4

Table 4.9

Ratio of LRU performance indices				
string	page fault rate (artificial/real)		space time product (artificial/real)	
	m=21	m=31	m=21	m=31
T00	148.63	14.78	139.88	13.03
T01	148.63	15.22	139.88	13.36
T02	148.63	15.07	139.88	13.28
T10	68.49	7.10	64.42	6.34
T11	68.49	7.49	64.42	6.67
T12	67.81	7.52	63.80	6.69
TT00	89.04	13.37	83.44	11.76
TT01	89.04	13.63	83.44	12.02
TT02	89.04	13.55	83.44	11.93
TT10	41.44	6.55	39.02	5.85
TT11	41.51	6.81	39.08	6.08
TT12	41.16	6.78	38.77	6.05

CHAPTER 5

Conclusions

5.1. Choices Among Available Options

Based on the outcome of the experiment, one of the available strategies to generate an artificial string can be chosen. An optimized version of the generation program, without statistics gathering, is expected to run twice as fast as the version we have used to gather statistics such as program profiles. The tradeoff among the possible choices has two aspects : accuracy and complexity. A more sophisticated strategy than the simplest one should definitely be selected if performance indices are unacceptably inaccurate without it. Such a strategy should also be incorporated into the generation algorithm if it adds very little complexity to the implementation and to the cost of the generation phase, but gives reasonable returns in the increased accuracies of some performance indices.

5.1.1. WS policy

Since the accuracy of the double T based generative model is practically independent of which strategies are chosen, it is clear that simplicity is the primary concern. TT00 could be a reasonable candidate; however, TT10 has a much smaller storage requirement due to its much higher coefficient of resilience. Single T generation is not considered here because of the clear advantage of double T generation in terms of stability and reliability.

5.1.2. PFF policy

Double T generation provided acceptable results even in the PFF policy case. It is clear that taking into account the number of distinct pages and using them in a FIFO order when a new page is needed is very cost-effective.

5.1.3. LRU policy

The results from LRU experiments were not very satisfactory, but we should not expect that a WS-based approach for string generation will automatically provide a good accuracy under LRU without requiring some guidance for generating LRU-oriented strings. Even in this case, double T generation with the minimum number of pages is the most cost-effective solution among those studied in this report.

5.1.4. Summary considerations

All things considered, a double T generation algorithm is undoubtedly a better choice than any single T generation algorithm. The coefficient of resilience can be incorporated to minimize storage requirements and improve the accuracy in a non-WS environment. At the same time, having the artificial string reference the same number of distinct pages as the real one is also very beneficial.

In summary, the TT11 strategy should be used. Various results for the artificial string TT11 can be found in Appendices F through I. Making a special effort to fit the string profile to that of the real string is not worthwhile with respect to the performance indices we are interested in.

5.2. Future Research

In order to improve the accuracy of the model in an LRU environment, incorporating into the algorithm the first order properties of LRU behavior may prove to be useful. For instance, including more than one stack distance probability, e.g., d_2 and d_3, in the generation phase of the algorithm may shape the artificial string to be more LRU-like.

With single T or double T generation of artificial strings, we could define an acceptable interval for a WS characterization as the interval of window sizes in which the indices measured under the WS policy are within 5 percent of the real string values. A major difficulty of an approach of this kind is that we do not know whether the deviation or error of the model as a function of the window size is concave or convex. The exhaustive exploration of the error curve is not a very attractive approach.

Along with the notion of the acceptable interval for a WS characterization, we could introduce a similar notion for the distributions of performance indices. For example, the Chi-square test at some level of significance could be used to compare the working set size distributions or the interfault time distributions.

The obvious extension of the double T generation approach is a triple T generation algorithm. With three reasonably spaced window sizes, it is plausible that not only the length of the acceptable interval will increase, but also the model's accuracy in terms of first moment results as well as of distributions. However, it is not known whether the further gains in model accuracy will justify the increase in the complexity of the generation algorithm.

BIBLIOGRAPHY

[Bril75a]
D. R. Brillinger, "Time Series : Data Analysis and Theory," Holt, Rinehart, and Winston, Inc., 1975.

[Chu 76a]
W. W. Chu and H. Opderbeck, "Program Behavior and the Page Fault Frequency Replacement Algorithm," Computer 9, November 1976, pp. 29-38.

[Denn68a]
P. J. Denning, "The Working Set Model of Program Behavior," CACM Vol. 11 No. 5, May 1968, pp. 323-333.

[Denn72a]
P. J. Denning and S. C. Schwartz, "Properties of the Working Set Model," CACM Vol. 15 No. 3, March 1972, pp. 191-198.

[Dutt81a]
C. R. Dutt, "Dynamic Characterization and Reproduction of Program Memory Consumption with Working-Set-Based Generative Model," Master's Report, University of California at Berkeley, August 1981.

[Ferr76a]
D. Ferrari, "The Improvement of Program Behavior," Computer 9, November 1976, pp. 39-47.

[Ferr81a]
D. Ferrari, "Characterization and Reproduction of the Referencing Dynamics of Programs," Performance '81, F. Kylstra, Ed., North-Holland, Amsterdam, November 1981, pp. 363-372.

[Ferr81b]
D. Ferrari, "A Generative Model of Working Set Dynamics," Proc. of the Sigmetrics Conference on Measurement and Modeling on Computer Systems, 1981, pp. 52-57.

[Ferr82a]
D. Ferrari, "Existence Theorem for 2-T Generation," private communication, January 1982.

[Lee 82a]
T. P. Lee, "Properties of Flat-faults in Working Set Model," PROGRES Report, Computer Science Division, University of California at Berkeley, to appear.

[Matt70a]
R. L. Mattson, J. Gescei, D. R., Slutz, and I. L. Traiger, "Evaluation Techniques for Storage Hierarchies," IBM Syst. J., 9, 2, 1970, pp. 79-117.

[Smit76a]
A. J. Smith, "A Modified Working Set Paging Algorithm," IEEE TC, C-25, September 1976, pp. 94-101.

[Spir77a]
J. R. Spirn, "Program Behavior : Models and Measurements," Elsevier, New York, 1977.

APPENDIX A

Theorem Proofs

Theorem 3.1

If S = $s_1 \cdots s_n$ is a wss string obeying the conditions of Theorem 2.1 for window size T, then the extended string $s_1 \cdots s_n s_n \cdots s_n$ of length $n+T$ satisfies the conditions of Theorem 2.1.

proof :

If we extend the wss sequence by adding T values all equal to that of the working set size at time n, the sequence we obtain is still a bounded positive continuous string with parameter T. This is trivial, since all three conditions in Definition 2.1 are satisfied. Specifically,

(i) $s_1=1$

(ii) $0<s_i \le T$ for $1 \le i \le n+T$

(iii) $|s_i - s_{i-1}| \le 1$ for $2 \le i \le n+T$

Furthermore, for $n+T \ge t > n$, d_t is 0 and therefore $d_t < s_t$ in this range also. By Theorem 2.1, this theorem is thus proved.

Theorem 3.2

Given two wss strings $S_1 = s_{1_1} \cdots s_{1_n}$ and $S_2 = s_{2_1} \cdots s_{2_n}$ obeying the conditions of Theorem 2.2 with window sizes T_1 and T_2 respectively, then there exists a a reference string R of length n having wss characterizations S_1 and S_2 with window sizes T_1 and T_2 respectively which is generated by the extended wss strings $s_{1_1} \cdots s_{1_n} s_{1_n} \cdots s_{1_n}$ of length $n+T_1$ and $s_{2_1} \cdots s_{2_n} s_{2_n} \cdots s_{2_n}$ of length $n+T_1$.

proof :

Each extended wss sequence individually satisfies the properties referred to in the proof of Theorem 3.1. The $S_1 \ge S_2$ relationship holds no matter how much longer both are extended. Now, let us verify the three properties in Definition 2.5.

Properties (i) and (ii) hold (furthermore, they need not be satisfied for $t>n$).

Property (iii) holds if for all t $s_{2_t} = s_{2_{t-1}} - 1$ and $s_{1_{t+T_1-T_2}} \ge s_{1_{t+T_1-T_2-1}}$ implies the existence of a unique $k \in (t, t+T_1-T_2]$, such that $s_{2_k} = s_{2_{k-1}} + 1$ and $s_{1_k} \le s_{1_{k-1}}$.

Property (iii) is not satisfied in general. In particular, it may not be satisfied when the wss characterization with the smaller window size has a decrease at some $t > n - T_1 + T_2$ while the extended wss characterization with the larger window size remains unchanged at $t + T_1 - T_2 > n$.

However, by examining the generation process more carefully, we could imagine that the condition implied in property (iii) is satisfied at time $k=t+T_1-T_2$ if there is no such unique $k\in(t,t+T_1-T_2]$, that is, if there is no increase in wss for S_2 at time k. In essence, we imagine that there is a wss increase for S_2 at time k. Notice that this could only occur at times greater than n because $k=t+T_1-T_2>n$. This slightly 'modified' extension rule has the same effect as the original extension rule to the page deadline requirement update process and the selection process for pages before time $n+1$. Only at times greater than n, there may exist problem in generating reference string with the original extension rule. For the purpose of generating n references by extending both wss characterizations by T_1 references with values all equal to those of the working set sizes at time n respectively, this extended scheme would suffice.

Before Theorem 4.1 is proved, it is necessary to introduce one definition and three lemmas.

Definition A.1

String S is said to be wss-equivalent to string R in interval I if it has the same working set size as R for any $T\in I$ and for all t.

Lemma A.1

The relation of wss-equivalence forms an equivalence class.

proof :

The reflexive and symmetric properties hold for obvious reasons. The transitive property holds since the equality relationship between working set sizes is transitive. Specifically,

$$w_1(T,t)=w_2(T,t) \text{ and } w_2(T,t)=w_3(T,t) \text{ implies } w_1(T,t)=w_3(T,t)$$

Lemma A.2

Let strings S and R be generated each from two wss characterizations with window sizes T_s and T_l, with $T_s<T_l$. Furthermore, let the algorithms by which S and R are generated differ only in their treatment of the references at the times when both wss characterizations contain wss increases. Then, strings S and R are wss-equivalent in $[1,T_l)$.

proof :

Without loss of generality, R is the string to which the S_i's are converging. We shall construct an arbitrary large number of wss-equivalent strings S_i' for $i\geq 0$ in interval $[1,T_l)$, where each string S_i has the same prefix of length i as R. Define S_0 = S. Furthermore, assume that $T\in[1,T_l)$, that is, $T<T_l$. The following proof is by induction.

S_0 has the same prefix of length 0 (null prefix) as R and since S and S_0 are the same, they are obviously wss-equivalent. Assume that S_k and is wss-equivalent to S_{k-1}; hence, by Lemma A.1, it will be wss-equivalent to all S_i's with $i<k$, including S.

If both the given wss characterizations are not increasing at time $k+1$, then the same page is selected for both S_k and R by assumption and because of the fact that they have the same prefix of length k. In this case, we set $S_{k+1} = S_k$, and S_{k+1} is obviously wss-equivalent to S_k.

If both the given wss characterizations are increasing, different new pages may be selected from the joint external queue. Assume that string R has a reference 'y' and string S_k has a reference 'x' at time $k+1$. If they happen to be the same, we can set $S_{k+1}=S_k$ as in the previous case. If 'x' is different from 'y', the new string S_{k+1} is constructed as follows : S_{k+1} is the same as S_k except that, for $t>k$ the page names 'x' and 'y' are interchanged. We have to show that S_{k+1} is wss-equivalent to S_k.

For $t \leq k$, $w_{k+1}(T,t)=w_k(T,t)$ since the strings have the same prefix.

For $t \geq k+T$, $w_{k+1}(T,t)=w_k(T,t)$ since the working set with window size T does not contain any page which was referenced only before or at k. Interchanging the names of 'x' and 'y' does not change the working set size.

For the $k+T>t>k$ case, we know that 'x' cannot appear anywhere between $k-T_l+1$ and k. Since $T<T_l$, 'x' cannot appear anywhere between $k-T+1$ and k. Now, before time $k+1$, no 'x' contributes to the working set with window size T at time t such that $k+T>t>k$. Therefore, interchanging the names 'x' and Whether 'x' appears after time $k+1$ is not important, because all 'x's will be changed to a new name.

Thus, we have completed the induction step. S_{k+1} is wss-equivalent all the way back to S and at the same time has the same prefix of length $k+1$ as R. For any given n, we can construct such S_n, and, by Definition A.1, R and S are wss-equivalent in $[1,T_l)$.

Lemma A.3

Let strings S and R be generated each from two wss characterizations with window sizes T_s and T_l, with $T_s<T_l$. Furthermore, let the algorithms by which S and R are generated differ only in their treatment of the references at the times when both wss characterizations do not increase. Then, strings S and R are wss-equivalent in (T_s,∞).

proof :

Without loss of generality, R is the string to which the S_i's are converging. We shall construct an arbitrary large number of wss-equivalent strings S_i' for $i \geq 0$ in interval (T_s,∞), where each string S_i has the same prefix of length i as R. Define S_0 = S. Furthermore, assume that $T \in (T_s,\infty)$, that is, $T>T_s$. The following proof is by induction.

S_0 has the same prefix of length 0 (null prefix) as R and since S and S_0 are the same, they are obviously wss-equivalent. Assume that S_k and is wss-equivalent to S_{k-1}; hence, by Lemma A.1, it will be wss-equivalent to all S_i's with $i<k$, including S.

If either the given wss characterizations increases at time $k+1$, then the same page is selected for both S_k and R by assumption and because of the fact that they have the same prefix of length k. In this case, we set $S_{k+1} = S_k$, and S_{k+1} is obviously wss-equivalent to S_k.

If both the given wss characterizations do not increase, different pages may be selected from the joint candidate queue. Assume that string R has a reference 'y' and string S_k has a reference 'x' at time $k+1$. If they happen to be the same, we can set $S_{k+1}=S_k$ as in the previous case. If 'x' is different from 'y', the new string S_{k+1} is constructed as follows : S_{k+1} is the same as S_k except that, for $t>k$ the page names 'x' and 'y' are interchanged. We have to show that S_{k+1} is wss-equivalent to S_k.

For $t \le k$, $w_{k+1}(T,t)=w_k(T,t)$ since the strings have the same prefix.

For $t \ge k+T$, $w_{k+1}(T,t)=w_k(T,t)$ since the working set with window size T does not contain any page which was referenced only before or at k. Interchanging the names of 'x' and 'y' does not change the working set size.

For the $k+T>t>k$ case, let us first show that $y \in W_k(T_s,t)$ for $k+T_s \ge t>k$. This is true because 'y' is a candidate page selected by T, but not by S_k at time $k+1$. Therefore, 'y' has to appear in any window of size T_s that covers time $k+1$. In other words, the page 'y' cannot be separated by more than T_s time. By assumption, $T>T_s$, we have containment relationship :

$$W_k(T_s,t) \subseteq W_k(T,t)$$

Therefore, for $k+T_s \ge t>k$, $y \in W_k(T,t)$. For $k+T>t>k+T_s$ case, we know that the window would cover from time $k+2$ to time $k+T_s-1$ in which 'y' has to show up because 'y' is a candidate page that is not referenced at time $k+1$ by S_k. Hence, we know for $k+T>t>k$, $y \in W_k(T,t)$. If 'y' is guaranteed to exist, and we know also that 'x' exists, interchanging these two names will not affect the working set size.

Thus, we have completed the induction step. S_{k+1} is wss-equivalent all the way back to S and at the same time has the same prefix of length $k+1$ as R. For any given n, we can construct such S_n, and, by Definition A.1, R and S are wss-equivalent in (T_s,∞).

Theorem 4.1

Strings TT00, TT01, TT02, TT10, TT11, TT12 generated from two wss characterizations with window sizes T_s and T_l, with $T_s<T_l$, have the same wss characterizations for any T such that $T_s<T<T_l$.

proof :

Algorithms that generate strings TT00 and TT10 satisfy the assumptions of Lemma A.3. Hence, strings TT00 and TT10 are wss-equivalent in (T_s,∞), in particular, in (T_s,T_l). Algorithms that generate strings TT00, TT01, and TT02 satisfy the assumptions of Lemma A.2. Hence, strings TT00, TT01, TT02 are, by Lemmas A.2 and A.1, wss-equivalent in $[1,T_l)$, in particular, in (T_s,T_l). A similar argument can be repeated to strings TT10, TT11, and TT12. Then, by Lemma A.1, the six strings are all wss-equivalent in (T_s, T_l).

APPENDIX B

Program Profile of the Real String

Total number of page references : 500000
Total number of changes : 227827
Coefficient of resilience : 0.544
Total number of distinct pages : 110

page no.	count	cumulative %	program profile
1304	125782	25.2	•••
1291	101775	45.5	•••
1246	38697	53.3	••••••••••••••••
1292	32379	59.7	•••••••••••••
1305	22364	64.2	•••••••••
1278	20333	68.3	•••••••••
1212	11885	70.6	•••••
1245	8722	72.4	••••
1244	8087	74.0	••••
1208	6842	75.4	•••
1259	5784	76.5	•••
1294	5662	77.7	•••
1211	5180	78.7	•••
1272	5025	79.7	••
1282	4613	80.6	••
1249	4349	81.5	••
1222	4320	82.4	••
1274	3984	83.2	••
1223	3950	83.9	••
1219	3861	84.7	••
1280	3692	85.5	••
1216	3531	86.2	••
1250	3352	86.8	••
1285	3188	87.5	••
1297	3045	88.1	••
1224	2757	88.6	••
1217	2470	89.1	•
1215	2390	89.6	•
1287	2349	90.1	•
1221	2263	90.5	•
1220	2255	91.0	•
1218	2233	91.4	•
			.
			.
10	1	100.0	•

APPENDIX C

WS Results for the Real String

```
STATISTICS FOR real.string
Total of 500000 references with window size 10000
Total no. of slope changes: 619
Space-Time product: 1.066510e+08

Mean working set size:  20.90
Maximum working set size: 56

                                        working set size distribution
ws-size        count      cumulative %
   1              1           0.0         •
   2              2           0.0         •
   3             17           0.0         •
   4              5           0.0         •
   5             98           0.0         •
   6           4903           1.0         ••
   7          29411           6.9         ••••••••••
   8           6405           8.2         ••
   9          20625          12.3         •••••••
  10          32886          18.9         •••••••••••
  11          14164          21.7         •••••
  12           9177          23.5         •••
  13           8746          25.3         •••
  14          10443          27.4         ••••
  15         160379          59.5         •••••••••••••••••••••••••••••••••••••••••••••••••••
  16           4686          60.4         ••
  17           7587          61.9         •••
  18           9295          63.8         •••
  19           2082          64.2         •
  20           2890          64.8         •
  21           1272          65.0         •
  22           1397          65.3         •
  23          10290          67.4         ••••
  24            387          67.4         •
  25           1952          67.8         •
  26           3322          68.5         ••
  27           2797          69.0         •
  28          13589          71.8         •••••
  29          16825          75.1         ••••••
  30           4207          76.0         ••
  31           6540          77.3         •••
```

```
                              working set size distribution
ws-size       count         cumulative %
  32          12039            79.7          ****
  33           9055            81.5          ***
  34           8371            83.2          ***
  35           9729            85.1          ****
  36          10199            87.2          ****
  37           2850            87.7          *
  38           2415            88.2          *
  39           6281            89.5          **
  40           7417            90.9          ***
  41           4160            91.8          **
  42           3654            92.5          **
  43           5741            93.7          **
  44           2748            94.2          *
  45           5251            95.3          **
  46           1350            95.5          *
  47           1750            95.9          *
  48           2672            96.4          *
  49           3226            97.1          **
  50           3514            97.8          **
  51           2569            98.3          *
  52           3613            99.0          **
  53           2499            99.5          *
  54           1671            99.8          *
  55            820           100.0          *
  56             26           100.0          *

Page fault rate: 6.480000e-04

Mean time between faults : 1543.21
Total page faults: 324
Maximum interfault time : 111695

                              Interfault time distribution
time          count         cumulative %
   0+          159             49.1          ***************************************************
 200+           62             68.2          ********************
 400+           13             72.2          *****
 600+           11             75.6          ****
 800+            9             78.4          ***
1000+            9             81.2          ***
1200+            6             83.0          **
1400+            3             84.0          *
1600+            2             84.6          *
1800+            3             85.5          *
2000+            4             86.7          **
2200+            0             86.7          *
2400+            2             87.3          *
```

Interfault time distribution

time	count	cumulative %	
2600+	4	88.6	••
2800+	3	89.5	•
3000+	3	90.4	•
3200+	2	91.0	•
3400+	1	91.4	•
3600+	0	91.4	•
3800+	4	92.6	••
4000+	0	92.6	•
4200+	2	93.2	•
4400+	1	93.5	•
4600+	3	94.4	•
4800+	0	94.4	•
5000+	1	94.8	•
5200+	1	95.1	•
5400+	1	95.4	•
5600+	3	96.3	•
5800+	0	96.3	•
6000+	0	96.3	•
6200+	0	96.3	•
6400+	0	96.3	•
6600+	0	96.3	•
6800+	0	96.3	•
7000+	0	96.3	•
7200+	0	96.3	•
7400+	0	96.3	•
7600+	1	96.6	•
7800+	0	96.6	•
8000+	0	96.6	•
8200+	0	96.6	•
8400+	0	96.6	•
8600+	1	96.9	•
8800+	0	96.9	•
9000+	0	96.9	•
9200+	0	96.9	•
9400+	0	96.9	•
9600+	0	96.9	•
9800+	10	100.0	••••

APPENDIX D

LRU Results for the Real String

```
STATISTICS FOR real.string
Total of 500000 references with 21 pages allocated in LRU stack
Space-time product: 1.633800e+08

Page fault rate: 1.456000e-03

Mean time between faults : 686.81
Total page faults: 728
Maximum interfault time : 111416

                               Interfault time distribution
time          count        cumulative %
   0+          495            68.0          •••••••••••••••••••••••••••••••••••••••••••••••••••
 200+           99            81.6          •••••••••••
 400+           51            88.6          ••••••
 600+           19            91.2          ••
 800+           15            93.3          ••
1000+           14            95.2          ••
1200+            7            96.2          •
1400+            5            96.8          •
1600+            5            97.5          •
1800+            5            98.2          •
2000+            2            98.5          •
2200+            1            98.6          •
2400+            0            98.6          •
2600+            1            98.8          •
2800+            0            98.8          •
3000+            0            98.8          •
3200+            0            98.8          •
3400+            1            98.9          •
3600+            1            99.0          •
3800+            0            99.0          •
4000+            0            99.0          •
4200+            0            99.0          •
4400+            0            99.0          •
4600+            0            99.0          •
4800+            0            99.0          •
5000+            0            99.0          •
5200+            0            99.0          •
5400+            0            99.0          •
5600+            0            99.0          •
5800+            0            99.0          •
6000+            0            99.0          •
6200+            0            99.0          •
6400+            0            99.0          •
6600+            0            99.0          •
6800+            0            99.0          •
7000+            0            99.0          •
7200+            0            99.0          •
7400+            0            99.0          •
```

```
                              Interfault time distribution
 time            count        cumulative %
 7600+             0              99.0          •
 7800+             0              99.0          •
 8000+             0              99.0          •
 8200+             0              99.0          •
 8400+             0              99.0          •
 8600+             0              99.0          •
 8800+             0              99.0          •
 9000+             0              99.0          •
 9200+             0              99.0          •
 9400+             0              99.0          •
 9600+             0              99.0          •
 9800+             7             100.0          •

                              stack distance distribution
 distance          count        cumulative %
    1             272173           54.4         •••••••••••••••••••••••••••••••••••••••••••••••••••
    2             155878           85.6         •••••••••••••••••••••••••••••
    3              41061           93.8         ••••••••
    4              12054           96.2         •••
    5               6690           97.6         ••
    6               4978           98.6         •
    7               1857           98.9         •
    8               1025           99.1         •
    9                721           99.3         •
   10                604           99.4         •
   11                382           99.5         •
   12                369           99.6         •
   13                238           99.6         •
   14                234           99.7         •
   15                315           99.7         •
   16                134           99.7         •
   17                126           99.8         •
   18                132           99.8         •
   19                 99           99.8         •
   20                112           99.8         •
   21                 90           99.9         •
   ∞                 728          100.0         •
```

APPENDIX E

PFF Results for the Real String

```
STATISTICS FOR real.string
Total of 500000 references with interfault threshold 1543
Space-Time product: 1.172574e+08
Total no. of slope changes: 417

Mean memory occupancy:  20.71
Maximum memory occupancy: 67

                                    working set size distribution
ws-size         count       cumulative %
    1               1           0.0          •
    2               2           0.0          •
    3              17           0.0          •
    4               5           0.0          •
    5              98           0.0          •
    6               3           0.0          •
    7           20794           4.2          •••••••••••••
    8           30024          10.2          ••••••••••••••••••
    9           36636          17.5          ••••••••••••••••••••••
   10           20145          21.5          ••••••••••••
   11            6800          22.9          •••••
   12            3794          23.7          •••
   13           16754          27.0          ••••••••••
   14           13533          29.7          •••••••••
   15           84027          46.5          ••••••••••••••••••••••••••••••••••••••••••••••••••
   16            8168          48.2          •••••
   17           43665          56.9          ••••••••••••••••••••••••••
   18             544          57.0          •
   19            1030          57.2          •
   20             692          57.3          •
   21            4139          58.2          •••
   22             772          58.3          •
   23           63348          71.0          ••••••••••••••••••••••••••••••••••••••
   24            4658          71.9          •••
   25            4719          72.9          •••
   26           11199          75.1          •••••••
   27           11638          77.4          •••••••
   28            6983          78.8          •••••
   29           17341          82.3          •••••••••••
   30            4127          83.1          •••
   31            5092          84.1          ••••
   32            6250          85.4          ••••
   33            7855          87.0          •••••
   34            4711          87.9          •••
   35            2278          88.4          ••
   36           12380          90.8          ••••••••
   37            3865          91.6          •••
   38             802          91.8          •
   39            2034          92.2          ••
   40            4440          93.1          •••
```

```
                              working set size distribution
ws-size        count       cumulative %
  41            3819           93.8            •••
  42            1376           94.1            •
  43             900           94.3            •
  44            2149           94.7            ••
  45            4259           95.6            •••
  46             731           95.7            •
  47             420           95.8            •
  48             210           95.8            •
  49            3567           96.6            •••
  50            1162           96.8            •
  51              81           96.8            •
  52              61           96.8            •
  53            6832           98.2            •••••
  54              12           98.2            •
  55             357           98.3            •
  56             420           98.3            •
  57             619           98.5            •
  58              44           98.5            •
  59            1026           98.7            •
  60             383           98.8            •
  61              30           98.8            •
  62              43           98.8            •
  63              90           98.8            •
  64               8           98.8            •
  65             557           98.9            •
  66              11           98.9            •
  67            5472          100.0            ••••

Page fault rate: 8.420000e-04

Mean time between faults : 1187.65
Total page faults: 421
Maximum interfault time : 80399

                              interfault time distribution
time           count       cumulative %
   0+           206           48.9             •••••••••••••••••••••••••••••••••••••••••••••••••••
 200+            77           67.2             •••••••••••••••••••
 400+            29           74.1             ••••••••
 600+            18           78.4             •••••
 800+            17           82.4             •••••
1000+             8           84.3             ••
1200+             6           85.7             ••
1400+             2           86.2             •
1600+             1           86.5             •
1800+             6           87.9             ••
2000+             5           89.1             ••
2200+             1           89.3             •
2400+             6           90.7             ••
2600+             7           92.4             ••
2800+             2           92.9             •
3000+             1           93.1             •
3200+             3           93.8             •
3400+             2           94.3             •
3600+             0           94.3             •
3800+             4           95.2             •
4000+             0           95.2             •
```

interfault time distribution

time	count	cumulative %	
4200+	2	95.7	*
4400+	0	95.7	*
4600+	1	96.0	*
4800+	1	96.2	*
5000+	0	96.2	*
5200+	1	96.4	*
5400+	1	96.7	*
5600+	2	97.1	*
5800+	0	97.1	*
6000+	0	97.1	*
6200+	0	97.1	*
6400+	0	97.1	*
6600+	0	97.1	*
6800+	1	97.4	*
7000+	0	97.4	*
7200+	1	97.6	*
7400+	0	97.6	*
7600+	0	97.6	*
7800+	0	97.6	*
8000+	0	97.6	*
8200+	0	97.6	*
8400+	0	97.6	*
8600+	1	97.9	*
8800+	0	97.9	*
9000+	0	97.9	*
9200+	0	97.9	*
9400+	0	97.9	*
9600+	0	97.9	*
9800+	9	100.0	***

APPENDIX F

Program Profile of String TT11

```
During string generation process,
Combined page select state distribution :
s0 : 498599 s1 : 309 s2 : 567 s3 : 0 s4 : 281 s5 : 0 s6 : 244 s7 : 0 s8 : 0
Combined page update state distribution :
s0 : 498646 s1 : 568 s2 : 313 s3 : 218 s4 : 0 s5 : 0 s6 : 0 s7 : 0 s8 : 255

Total number of page references : 500000
Total number of changes : 228137
Coefficient of resilience :  0.544

                                              program profile
page no.        count       cumulative %
    1            6385            1.3          ••••••••••
    2            3131            1.9          •••••
    3           16301            5.2          •••••••••••••••••••••••••
    4             628            5.3          •
    5           10418            7.4          ••••••••••••••••
    6            2855            7.9          •••••
    7            2552            8.5          ••••
    8            1045            8.7          ••
    9             496            8.8          •
   10            2871            9.3          •••••
   11             969            9.5          ••
   12            1166            9.8          ••
   13            1798           10.1          •••
   14            1074           10.3          ••
   15            2689           10.9          •••••
   16             784           11.0          ••
   17            1425           11.3          •••
   18           13820           14.1          •••••••••••••••••••••
   19            2168           14.5          ••••
   20             891           14.7          ••
   21            4974           15.6          ••••••••
   22            2704           16.2          •••••
   23            2507           16.7          ••••
   24              54           16.7          •
   25            2869           17.3          •••••
   26           10855           19.4          •••••••••••••••••
   27            2679           20.0          •••••
   28           13788           22.7          •••••••••••••••••••••
   29           14804           25.7          •••••••••••••••••••••••
   30             983           25.9          ••
   31            1020           26.1          ••
   32             522           26.2          •
   33            2049           26.6          ••••
   34            1408           26.9          •••
   35            1469           27.2          •••
   36            3927           28.0          ••••••
   37             518           28.1          •
   38             482           28.2          •
```

program profile

page no.	count	cumulative %	
39	622	28.3	•
40	1873	28.7	•••
41	1721	29.0	•••
42	2705	29.6	•••••
43	826	29.7	••
44	28	29.7	•
45	11456	32.0	••••••••••••••••••
46	2673	32.6	•••••
47	1185	32.8	••
48	797	33.0	••
49	4411	33.8	•••••••
50	16624	37.2	••••••••••••••••••••••••••
51	13421	39.8	•••••••••••••••••••••
52	1976	40.2	••••
53	9235	42.1	•••••••••••••••
54	743	42.2	••
55	12227	44.7	•••••••••••••••••••
56	2691	45.2	•••••
57	728	45.4	••
58	1155	45.6	••
59	1840	46.0	•••
60	1097	46.2	••
61	4006	47.0	•••••••
62	2664	47.5	•••••
63	2597	48.0	••••
64	5962	49.2	••••••••••
65	13633	52.0	•••••••••••••••••••••
66	649	52.1	•
67	3766	52.8	••••••
68	115	52.9	•
69	1558	53.2	•••
70	1598	53.5	•••
71	504	53.6	•
72	1097	53.8	••
73	169	53.8	•
74	1850	54.2	•••
75	1357	54.5	•••
76	12561	57.0	••••••••••••••••••••
77	5892	58.2	•••••••••
78	5096	59.2	••••••••
79	13219	61.8	•••••••••••••••••••••
80	5444	62.9	•••••••••
81	10509	65.0	••••••••••••••••
82	7932	66.6	•••••••••••••
83	1442	66.9	•••
84	5738	68.1	•••••••••
85	3171	68.7	•••••
86	1232	68.9	••
87	5871	70.1	•••••••••
88	11533	72.4	••••••••••••••••••
89	544	72.5	•
90	140	72.6	•
91	9474	74.4	•••••••••••••••
92	12877	77.0	••••••••••••••••••••
93	803	77.2	••
94	9475	79.1	•••••••••••••••
95	7907	80.7	•••••••••••••
96	3368	81.3	••••••
97	3918	82.1	••••••

```
                                                    program profile
page no.        count       cumulative %
   98           32927          86.7           ****************************************************
   99            5332          89.8           *********
  100            5225          90.8           ********
  101            4087          91.6           *******
  102             695          91.8           **
  103           16438          95.1           *************************
  104             531          95.2           *
  105            4848          96.1           ********
  106            2810          96.7           *****
  107             147          96.7           *
  108           11527          99.0           ******************
  109            1928          99.4           ***
  110            2922         100.0           *****
```

APPENDIX G

WS Results for String TT11 at T=10000

```
STATISTICS FOR /xtra/tplee/TT11
Total of 500000 references with window size 10000
Total no. of slope changes: 613
Space-Time product: 1.052375e+08

Mean working set size:  21.05
Maximum working set size: 56

                                      working set size distribution
ws-size      count       cumulative %
   1             1            0.0         *
   2             2            0.0         *
   3            17            0.0         *
   4             5            0.0         *
   5            98            0.0         *
   6          4903            1.0         **
   7         25417            6.1         ********
   8          7889            7.7         ***
   9         20054           11.7         *******
  10         33756           18.4         ***********
  11         13139           21.1         *****
  12          8104           22.7         ***
  13         11421           25.0         ****
  14          9479           26.9         ***
  15        159618           58.8         ***************************************************
  16          4953           59.8         **
  17          8550           61.5         ***
  18         11424           63.8         ****
  19           795           63.9         *
  20          4177           64.8         **
  21          1272           65.0         *
  22          1397           65.3         *
  23         10290           67.4         ****
  24           387           67.4         *
  25          1952           67.8         *
  26          2655           68.4         *
  27          1865           68.7         *
  28          9277           70.6         ***
  29         20552           74.7         *******
  30          5659           75.8         **
  31          6426           77.1         ***
  32          6488           78.4         ***
  33         12649           80.9         ****
  34         10388           83.0         ****
  35          7245           84.5         ***
  36          9207           86.3         ***
  37          2494           86.8         *
  38          2002           87.2         *
  39         11008           89.4         ****
  40          7479           90.9         ***
```

```
                                    working set size distribution
ws-size        count        cumulative %
  41            4218            91.7           ••
  42            3742            92.5           ••
  43            4636            93.4           ••
  44            2801            94.0           •
  45            6399            95.3           •••
  46            1350            95.5           •
  47            1750            95.9           •
  48            2872            96.4           •
  49            3226            97.1           ••
  50            3514            97.8           ••
  51            2158            98.2           •
  52            4024            99.0           ••
  53            2499            99.5           •
  54            1355            99.8           •
  55            1136           100.0           •
  56              26           100.0           •

Page fault rate: 6.420000e-04

Mean time between faults : 1557.63
Total page faults: 321
Maximum interfault time : 111695

                                    interfault time distribution
 time          count        cumulative %
   0+           158             49.2           •••••••••••••••••••••••••••••••••••••••••••••••••••
  200+           60             67.9           •••••••••••••••••••
  400+           15             72.6           •••••
  600+           13             76.6           •••••
  800+            7             78.8           •••
 1000+           11             82.2           ••••
 1200+            4             83.5           ••
 1400+            3             84.4           •
 1600+            3             85.4           •
 1800+            4             86.6           ••
 2000+            1             86.9           •
 2200+            0             86.9           •
 2400+            1             87.2           •
 2600+            4             88.5           ••
 2800+            3             89.4           •
 3000+            1             89.7           •
 3200+            1             90.0           •
 3400+            3             91.0           •
 3600+            0             91.0           •
 3800+            3             91.9           •
 4000+            1             92.2           •
 4200+            3             93.1           •
 4400+            1             93.5           •
 4600+            2             94.1           •
 4800+            1             94.4           •
 5000+            0             94.4           •
 5200+            2             95.0           •
 5400+            1             95.3           •
 5600+            1             95.6           •
 5800+            0             95.6           •
 6000+            0             95.6           •
 6200+            0             95.6           •
```

interfault time distribution

time	count	cumulative %	
6400+	0	95.6	•
6600+	0	95.6	•
6800+	2	96.3	•
7000+	0	96.3	•
7200+	0	96.3	•
7400+	0	96.3	•
7600+	0	96.3	•
7800+	0	96.3	•
8000+	0	96.3	•
8200+	0	96.3	•
8400+	0	96.3	•
8600+	1	96.6	•
8800+	0	96.6	•
9000+	0	96.6	•
9200+	0	96.6	•
9400+	0	96.6	•
9600+	0	96.6	•
9800+	11	100.0	••••

APPENDIX H

LRU Results for String TT11

```
STATISTICS FOR /xtra/tplee/TT11
Total of 500000 references with 21 pages allocated in LRU stack
Space-time product: 6.370350e+09

Page fault rate: 6.057000e-02

Mean time between faults :  16.51
Total page faults: 30285
Maximum interfault time : 111563

                                  interfault time distribution
time        count       cumulative %
   0+       30123           99.5          **************************************************
 200+          67           99.7          *
 400+          17           99.7          *
 600+          12           99.8          *
 800+          10           99.8          *
1000+           7           99.8          *
1200+           6           99.9          *
1400+           5           99.9          *
1600+           5           99.9          *
1800+           5           99.9          *
2000+           3           99.9          *
2200+           1           99.9          *
2400+           1           99.9          *
2600+           2           99.9          *
2800+           1           99.9          *
3000+           2           99.9          *
3200+           0           99.9          *
3400+           0           99.9          *
3600+           1           99.9          *
3800+           2          100.0          *
4000+           0          100.0          *
4200+           0          100.0          *
4400+           0          100.0          *
4600+           1          100.0          *
4800+           2          100.0          *
5000+           0          100.0          *
5200+           1          100.0          *
5400+           0          100.0          *
5600+           2          100.0          *
5800+           0          100.0          *
6000+           0          100.0          *
6200+           0          100.0          *
6400+           0          100.0          *
6600+           0          100.0          *
6800+           0          100.0          *
7000+           0          100.0          *
7200+           0          100.0          *
7400+           0          100.0          *
```

```
                                 interfault time distribution
time          count          cumulative %
7600+           0               100.0              •
7800+           0               100.0              •
8000+           0               100.0              •
8200+           0               100.0              •
8400+           0               100.0              •
8600+           1               100.0              •
8800+           0               100.0              •
9000+           0               100.0              •
9200+           0               100.0              •
9400+           0               100.0              •
9600+           0               100.0              •
9800+           8               100.0              •

                                  stack distance distribution
distance          count          cumulative %
    1            271863             54.4         ••••••••••••••••••••••••••••••••••••••••••••••••••••••
    2              3423             55.1         •
    3               713             55.2         •
    4              1440             55.5         •
    5             10968             57.7         •••
    6             67276             71.1         •••••••••••••
    7             34816             78.1         •••••••
    8             12046             80.5         •••
    9              6954             81.9         ••
   10              5846             83.1         ••
   11              2987             83.7         •
   12              1349             83.9         •
   13              3650             84.7         •
   14              8886             86.4         ••
   15             13055             89.1         •••
   16              1874             89.4         •
   17              3221             90.1         •
   18              2727             90.6         •
   19              3934             91.4         •
   20              4046             92.2         •
   21              8641             93.9         ••
   ∞              30285            100.0         ••••••
```

APPENDIX I

PFF Results for String TT11

```
STATISTICS FOR /xtra/tplee/TT11
Total of 500000 references with interfault threshold 1543
Space-Time product: 1.183963e+08
Total no. of slope changes: 413

Mean memory occupancy:  20.63
Maximum memory occupancy: 67

                                   working set size distribution
ws-size      count      cumulative %
   1             1          0.0       *
   2             2          0.0       *
   3            17          0.0       *
   4             5          0.0       *
   5            98          0.0       *
   6             3          0.0       *
   7         20794          4.2       *************
   8         30416         10.3       *******************
   9         36244         17.5       **********************
  10         22451         22.0       **************
  11          7402         23.5       *****
  12          5512         24.6       ****
  13         14113         27.4       *********
  14         11161         29.6       *******
  15         83607         46.4       **************************************************
  16          8604         48.1       ******
  17         43646         56.8       ***************************
  18           316         56.9       *
  19           402         57.0       *
  20           591         57.1       *
  21          3585         57.8       ***
  22           625         57.9       *
  23         63396         70.6       **************************************
  24          7977         72.2       *****
  25          5626         73.3       ****
  26         13444         76.0       *********
  27         12875         78.6       ********
  28         15311         81.6       **********
  29          4807         82.6       ***
  30          2536         83.1       **
  31          4444         84.0       ***
  32          4475         84.9       ***
  33         12438         87.4       ********
  34          4602         88.3       ***
  35          1850         88.7       **
  36         13004         91.3       ********
  37          3108         91.9       **
  38           800         92.1       *
  39          1465         92.4       *
  40          6278         93.6       ****
```

```
                                    working set size distribution
ws-size         count        cumulative %
  41            1149             93.8          •
  42            1376             94.1          •
  43             900             94.3          •
  44            2149             94.7          ••
  45            4259             95.6          •••
  46             731             95.7          •
  47             420             95.8          •
  48             210             95.8          •
  49            3567             96.6          •••
  50            1162             96.8          •
  51              81             96.8          •
  52              61             96.8          •
  53            6832             98.2          •••••
  54              12             98.2          •
  55             357             98.3          •
  56             420             98.3          •
  57             619             98.5          •
  58              44             98.5          •
  59            1026             98.7          •
  60             383             98.8          •
  61              30             98.8          •
  62              43             98.8          •
  63              90             98.8          •
  64               8             98.8          •
  65             557             98.9          •
  66              11             98.9          •
  67            5472            100.0          ••••

Page fault rate: 8.460000e-04

Mean time between faults : 1179.25
Total page faults: 424
Maximum interfault time : 80399

                                    interfault time distribution
time            count        cumulative %
   0+             203            47.9          •••••••••••••••••••••••••••••••••••••••••••••••••••
 200+              82            67.2          •••••••••••••••••••••
 400+              25            73.1          •••••••
 600+              15            76.7          ••••
 800+              14            80.0          ••••
1000+               9            82.1          •••
1200+               6            83.5          ••
1400+               7            85.1          ••
1600+               3            85.8          •
1800+               7            87.5          ••
2000+               4            88.4          •
2200+               3            89.2          •
2400+               8            91.0          ••
2600+               7            92.7          ••
2800+               2            93.2          •
3000+               1            93.4          •
3200+               3            94.1          •
3400+               3            94.8          •
3600+               0            94.8          •
3800+               3            95.5          •
4000+               0            95.5          •
```

interfault time distribution

time	count	cumulative %	
4200+	2	96.0	*
4400+	0	96.0	*
4600+	1	96.2	*
4800+	2	96.7	*
5000+	1	96.9	*
5200+	1	97.2	*
5400+	1	97.4	*
5600+	1	97.6	*
5800+	0	97.6	*
6000+	0	97.6	*
6200+	0	97.6	*
6400+	0	97.6	*
6600+	0	97.6	*
6800+	1	97.9	*
7000+	0	97.9	*
7200+	0	97.9	*
7400+	0	97.9	*
7600+	0	97.9	*
7800+	0	97.9	*
8000+	0	97.9	*
8200+	0	97.9	*
8400+	0	97.9	*
8600+	1	98.1	*
8800+	0	98.1	*
9000+	0	98.1	*
9200+	0	98.1	*
9400+	0	98.1	*
9600+	0	98.1	*
9800+	8	100.0	**