SU-SEL 75-037 SU-326P.39-2

THE STACK WORKING SET: A CHARACTERIZATION OF SPATIAL LOCALITY

by

B. Ramakrishna Rau

July 1975

Technical Report No. 95

The work described herein was supported by the U.S. Energy Research and Development Agency (formally AEC) under contract AT(04-3)326P.A.39. Computer time was made available by the Stanford Linear Accelerator Center (SLAC).

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY · STANFORD, CALIFORNIA

SU-SEL 75-037 SU-326P.39-2

THE STACK WORKING SET:
A CHARACTERIZATION OF SPATIAL LOCALITY

by

B. Ramakrishna Rau

July 1975

Technical Report No. 95

DIGITAL SYSTEMS LABORATORY
Stanford Electronics Laboratories
Stanford University
Stanford, California

The work described herein was supported by the U.S. Energy Research and Development Agency (formally AEC) under contract AT(04-3)326P.A.39. Computer time was made available by the Stanford Linear Accelerator Center (SLAC).

Digital Systems Laboratory
Stanford Electronics Laboratories

Technical Report No. 95

July 1975

THE STACK WORKING SET:
A CHARACTERIZATION OF SPATIAL LOCALITY

by

B. Ramakrishna Rau

ABSTRACT

Multilevel memory hierarchies are attractive from the point of view of cost-performance. However, they present far greater problems than two-level hierarchies when it comes to analytic performance evaluation. This may be attributed to two factors: firstly, the page size (or the unit of information transfer between two levels) varies with the level in the hierarchy, and, secondly, the request streams that the lower (slower) levels see are the fault streams out of the immediately higher levels. Therefore, the request stream seen by each level is not necessarily the same as the one generated by the processor. Since the performance depends directly upon the properties of the request stream, this poses a problem.

A model for program behavior, which explicitly characterizes the spatial locality of the program, is proposed and validated. It is shown that the spatial locality of a program is an invariant of the hierarchy when characterized in this manner. This invariance is used to solve the first problem stated--that of the varying page sizes. An approximate technique is advanced for the characterization of the fault stream as a function of the request stream and the capacity of the level. A procedure is then outlined for evaluating the performance of a multilevel hierarchy analytically.

I. INTRODUCTION

A sizeable body of literature exists in the areas of memory performance analysis and the modelling of program behavior . However, with but a few exceptions [MATT70, SLUT72, GECS74] all extant techniques are suitable only for use with two-level memory structures (Fig.1). In such a memory system, all information that is not available at the higher (faster) level is assumed to be available at the lower level. This type of structure accurately reflected most of the earlier computer system, and, for that matter, a majority of the present ones. However, with the growing disparity between the speeds of processor and memory technology, it is reasonable to assune that multi-level memory hierarchies will become increasingly popular. In a multi-level hierarchy, each level is slower and larger than the level immediately above it. If an item of information is not found at a particular level, the request will be handed down to the next, lower level, which, in turn, if unable to provide the item, will hand it down to the next level (Fig.2). In such a hierarchy, each level will see a different request stream. Accordingly, each level will, generally, manage itself independently of the others.

Such a configuration may be expected to be cost-effective if algorithms can be devised to manage the hierarchy, so that, at any instant, the information, most likely to be referenced, is present at the highest level. The second level should, ideally, contain that information which has the greatest probability of being requested by the highest level, i.e. the information requested by the processor but not present in the first level.

However, multi-level hierarchies present problems of greater complexity when it comes to performance analysis. The performance of a memory hierarchy may be evaluated either by simulation or by analysis. Simulation is the more accurate technique but is very expensive in computer time since it requires a new simulation run for every combination of parameters. Analytic techniques can be more flexible if the models used contain the necessary information. Herein lies the crux of the problem in relation to multi-level hierarchies. Before elaborating on this point further, it is worthwhile to examine the structure of existing models and to outline the requirements for models suitable for multi-level analysis.

II. MODELS OF PROGRAM BEHAVIOR

Analytic methods hinge upon the existence of two models:

1. A model for program behavior, and
2. A model for the structure of the memory.

The problem of modelling the structure of the memory will not be considered here. A fully associative structure will be assumed throughout for for every level.

The entire behavior of the program in question is contained in the string of references that it makes to the memory in the course of its execution. The entire string could, conceivably, be considered to be the model (and this is the case with simulation). A successful node1 for analytic purposes manages to condense the volume of statistics without losing the necessary information. Models of program behavior differ in their assumptions of what is relevant,

It is now, generally, agreed that the good models are those that capture that aspect of program behavior known as "locality" [DENN68, DENN70]. Locality is the tendency of programs to concentrate their references over any given interval of time, to a subset of the paces of the program, and for this subset to change rather slowly with time. As a consequence of this observation, two rather closely related

models, the Least-Recently-Used Stack Model [SHEM66, COFF73] and the Working Set Model [DENN68, DENN72], have been proposed. Both are based on the assumption that the more recently that a page has been referenced the greater is the probability that the next reference will be to it. If the probabilty of reference is a sharply decreasing function of time since last reference, it is easy to see that references tend to get concentrated in the small subset of the most recently referenced pages. Thus, the property of locality has been captured. The two models differ only in the method of specifying the reference probability as a function of the "recency" of last reference. Whereas in the Working Set Model (WSM) the probability is given as a function of time since last reference, in the Least-Recently-Used Stack Model (LRUSM), the probability is given as a function of the number of distinct references since the last reference to the page in question. The former is more suitable for analyzing the fault rate with variable memory allocation and the latter is more suited to analyzing fixed memory allocation policies. Both models are capable of predicting the fault rate for any given memory allocation (average allocation in the case of the WSM and fixed allocation in the case of the LRUSM). More recently, a nodified version 0-f the Independent Reference Ilodel, which demonstrates locality, has been proposed and validated [BASK75].

All these models are well suited to the two-level case, where the page size, i.e., the unit of transfer between the two levels, is fixed, but they are unable to cope with nuiti-level hierarchies in which the unit of transfer varies as one descends in the hierarchy. The reason for this lies in the fact that no information has been retained about the relationship between co-pages (i.e. pages which belong to the same larger page). We shall consider this problem in greater detail in the next section.

III. PROPERTIES OF A MODEL FOR MULTI-LEVEL HIERARCHIES

On re-examining the definition of locality, upon which these models are based, we see that a page size has been implicitly assumed. Only by so doing, can we in the LRUSM speak of the number of distinct pages referenced. Obviously, this number will depend upon the size of the pages. Again, in the WSM, the probability of referencing a page will depend upon its size.

A model should be able to predict the effect upon performance of changing the page size. Intuitively, a larger page size is advantageous if both halves of the page tend to get referenced fairly close together in time. If this is not the case, it is preferable to stay with half the page size, since fetching a larger page corresponds to fetching two pages of half the size, one of which is not used during the period of its residency in the memory level. It can be seen that the efficacy of a large page size depends upon the time scale -- in this case the period of residency of the page in the level before being displaced. The property of locality in the context of variable page sizes is rather confusing, and it can give rise to anomalous behavior [HATF72].

It has been suggested that the property of locality be split into two components -- temporal locality and spatial locality [MADN72]. Temporal locality is the property whereby a high probability exists that a page, once referenced, will be referenced shortly thereafter. Spatial locality, on the other hand, is said to exist if there is a high probability that pages close to the page currently being referenced, will be referenced within a short time. A high degree of temporal locality indicates a low fault rate with the LRU or WS replacement algorithms. A high degree of spatial locality suggests that doubling the page size might improve paging performance. These two properties may be defined more formally as follows:

The Temporal Locality of a program is characterized by the probability function F(t) where

F(t) = Prob[page i is referenced at time x+t | page i was last referenced at time x].

The Spatial Locality of a program is characterized by the probability function G(d,t) where

G(d,t) = Prob[page i+d is referenced at time x+t | page i was last referenced at time x].

These definitions can be used to construct a model of program behavior. The node1 consists of the two funtions

F(t) and G(d,t). (It may be noted that F(t) is nothing other than G(0,t)). However, this model requires an extremely large number of measurements and is analytically intractable. The fact of the matter is that it contains too much information to permit easy manipulation. Since we are, generally, interested in just a subset of all possible page sizes (powers of 2), it should be possible to do away with much of this information without compromising the utility of the model.

In the above definition of temporal locality, F(t) for a particular page size is dependent upon the temporal and spatial properties of the program for half the page size. When evaluating the performance of a hierarchy, this is undesirable. The temporal locality of a program will, generally, vary with the level in the hierarchy. An item of information that has just been referenced in a particular level is very unlikely to be referenced again in a short time even though the program makes repeated references to it. This is because a copy of that information exists at a higher level and, consequently, references will not filter down to the lower level. As a result, the spatial locality of the program would have to be measured at every level. It would be far better if the model of spatial locality were such that it was a function of only the program and not of the level in the hierarchy. In other words, the spatial

locality should be modelled such that it is an invariant in the hierarchy.

The temporal locality, on the other hand, will, most definitely, be a function of the level in the hierarchy. It should not, however, be characterized so as to be a function of the spatial locality.

Accordingly, the temporal locality must be characterized in terms of the smallest unit of information that the processor is capable of requesting. This might be larger than the smallest addressable unit. This is the "natural" choice by default, a smaller page size being meaningless. The spatial locality must be characterized by some property which is a function of page size, but is invariant with the level in the hierarchy.

Once this has been done, there is but one problem that remains to be solved. As has been noted, the fault stream out of one level (the request stream for the lower level) is different from the request stream to the level. The difference is mainly in the temporal locality of the two streams. A method must exist for deriving the model for the fault stream, given the model for the request stream and the parameters of the memory level.

We are now in a position to specify the properties of a model suitable for analyzing a multilevel hierarchy:

1. It should characterize the temporal locality of a program in a form that is tractable for analysis of fault rates and do so in such a manner that it is independent of the spatial locality of the program.

2. It should characterize the spatial locality of the program in a form that permits easy transformations from one page size to another and should do so in terms of a property that is invariant in the hierarchy.

3. The model should be capable of characterizing the fault stream given the parameters of the level under consideration.

4. The model should be flexible and permit the evaluation of hierarchies employing all types of memory management policies.

. THE STACK WORKING SET MODEL

The proposed model meets the first two requirements quite satisfactorily. It is not as successful in predicting the fault stream. The model is almost totally inflexible in that it is only able to analyze hierarchies that use the LRU or the WS replacement policies. In the context of fixed allocation policies, this is not a very great handicap since LRU is about the best, implementable algorithm. Many systems use either LRU or some variant. The same might be said for the WS replacement policy in the context of variable allocation. In its current version, the proposed model assumes that a store-through policy is used, i.e., every store request ripples all the way down the hierarchy, updating all copies of the information. Consequently, a displaced page need not be written back and can, merely, be deleted.

The LRU Stack Model is the best choice for a model if the memory management policy is demand-fetch and LRU replacement. The LRUSM is obtained by making measurements on the reference string of the program and by recording the number of times the referenced block was in each position of the LRU stack [MATT70]. By normalizing these counts with respect to the total number of references made by the program, we obtain the probability that the referenced page

is at a particular position in the stack. Let P(l;b) be the probability that the referenced page (of size b bytes) is found at position 1 in the stack. The position in the stack is, normally, referred to as the distance in the stack. The distance, 1, will be measured in units of a byte, and is the minimun size of a level (in bytes) that can contain the referenced page and all the pages that are above it in the LRU stack. (Note that in the literature, 1 is measured in units of a page. In our case this would not be appropriate, since the page size is a variable and cannot serve as a common yardstick).

The fault rate in a memory of size 1 is given by the probability that the current reference is to a page that is at a distance greater than 1. Thus, the fault rate

$$M(l;b) = 1 - \sum_{i=1}^{l} P(i;b) \qquad (1)$$

The assunptions made by the LRUSM are that the probability that the next reference will be to position 1 is equal to P(l;b), is independent of the previous requests and is independent of which page is currently in position 1. This last assumption is equivalent to the assumption that all pages are statistically identical in their behavior. As we shall see, this limits the flexibility of the model, for,

in fact, pages do not behave identically. Replacing the individualistic behavior of each page by an average behavior results in a loss of information. However, since the fault rate of a program, as measured over a large interval of tine, is itself an average behavior, the LRUSM is accurate in this respect. But it is totally unable to predict the dynamic behavior of the program.

We see that the LRUSM is a good predictor of temporal locality. But the LRUSM for any one page size is unable to predict the fault rate for any other page size, since no information has been retained, for each page, about the behavior of its co-page (buddy). One might suspect that the set of LRU statistics for all the page sizes of interest, would contain the necessary information. This is, in fact, the case. However, since the LRU statistics for any page size are a function of the tenporal locality, as well as of the spatial locality, they are not invariant in the hierarchy. The spatial locality has as yet to be extracted in a form independent of the temporal locality.

We first introduce the concept of the Stack Working Set. Let us consider a program which has built up a working set of n pages. Under the assumptions of the LRUSM, the expected time to the next fault is $1/M(nb;b)$. Time will, throughout, be measured in terms of the number of memory

references made by the processor. Therefore, the average time it takes to build up a working set of n+1 pages from the time that it had a working set of n pages is 1/M(nb;b). The time taken for a program to build up a working set of n pages, assuming it started with a null working set, is then given by

$$T(nb;b) = \sum_{i=0}^{n-1} \frac{1}{M(ib;b)} \qquad (2)$$

where M(0;b) = 1. (Figs.3,4,5)

T(l;b) is a function which exists only for values of l such that l = nb where n is an integer. Let us assume that T(l;b) is defined for non-integer values of n by suitable interpolation. The inverse function L(t;b) can then be defined such that

$$T(L(t;b)) = t \text{ and } L(T(l;b)) = l, \qquad (3)$$

where t is the time taken to build up the working set (Fig.6).

L(t;b) is, in some sense, analogous to the working set that would be built up by Denning's working set model with a

window size of t. However, being derived from the LRU statistics, it is not identical. Denning's WSM gives the average working set built up in time t, whereas L(t;b) relates the average time t required to build up a working set of a given size. Since L(t;b) is obtained from the LRU Stack model, this will be referred to as the Stack Working Set. Given the function L(t;b), T(1;b) can be obtained, by inverse interpolation. Having obtained T(1;b), 1/M(1;b) and, hence, M(1;b) can be deduced.

Spatial locality is characterized by formalizing the concept of the memory "wastage" incurred in increasing the page size. Accordingly, the spatial locality of a program is characterized by the function

$S(1;b_1,b_2)$ which is defined such that

$$L(t;b_2) = S(\ L(t;b_1);b_1,b_2\ ) \qquad (4)$$

where b_1,b_2 are the page sizes, and $b_1<b_2$.

$S(1;b_1,b_2)$ indicates the average size of the working set built up with a page size of b_2 as a function of the average size of the working set built up over the same period of time with a block size of b_1.

Based on the intuitive notion that the amount of "wastage" incurred in going from one page size to another is a function of only the program and is, therefore, invariant in the hierarchy, we hypothesize that $S(1;b_1,b_2)$ is an invariant in the hierarchy.

If the hypothesis is correct, since $S(1;b_1,b_2)$ is invariant we can, given $M(1;b_1)$ at any level, calculate $M(1;b_2)$ for the same level. $S(1;b_1,b_2)$ for any level can be obtained from the set of LRU statistics for all the page sizes of interest at level 1. Thus, the complete set of LRU statistics (for all page sizes) for a program do, indeed, contain all the information necessary to characterize both the temporal and spatial locality of the program.

The hypothesis regarding the invariance of the spatial locality function, $S(1;b_1,b_2)$, remains to be validated. This is best done by predicting $M(1;b_2)$ analytically and comparing it with the measured function. The computation involved is, however, quite laborious. A more elegant technique will first be derived and this will be used to validate the hypothesis.

THEOREM 1. Under the assumption of an invariant spatial locality function, the fault rate functions for the request streams to levels 1 and k, and for page sizes b_1 and b_2, are related by the following equation:

$$\frac{M_k(l_2;b_2)}{M_1(l_2;b_2)} = \frac{M_k(l_1;b_1)}{M_1(l_1;b_1)}$$

PROOF: Let $l_1 = L(t;b_1)$ (5)

and $l_2 = L(t;b_2)$. (6)

Then, by Eqns.3,4,

$T(l_1;b_1) = t$, (7)

$T(l_2;b_2) = t$ and (8)

$l_2 = S(l_1;b_1,b_2)$ (9)

Differentiating on both sides with respect to t we get,

$$\frac{dl_2}{dt} = \frac{dS(l_1;b_1,b_2)}{dl_1} * \frac{dl_1}{dt} \quad (10)$$

But $t = T(l_1;b_1) = T(l_2;b_2)$ and therefore,

$$\frac{dl_2}{dt} = \frac{dl_2}{dT(l_2;b_2)} \quad \text{and,} \quad (11)$$

$$\frac{dl_1}{dt} = \frac{dl_1}{dT(l_1;b_1)} \quad . \tag{12}$$

Since $dT(l_1;b_1)/dl_1 = 1/(M(l_1;b_1)*b_1)$ and (13)

$$dt(l_2;b_2)/dl_2 = 1/(M(l_2;b_2)*b_2) \tag{14}$$

we obtain the relation

$$M(l_2;b_2) = M(l_1;b_1) * \frac{dS(l_1;b_1,b_2)}{dl_1} * \frac{b_1}{b_2} \tag{15}$$

or,

$$\frac{M(l_2;b_2)}{M(l_1;b_1)} = \frac{dS(l_1;b_1,b_2)}{dl_1} * \frac{b_1}{b_2} \tag{16}$$

At this point we introduce some additional nomenclature. Let $M_k(l;b)$ be the fault probability function for the request stream to the k-th level in the hierarchy, and for a page size of b. Then, if the spatial locality function is, indeed, an invariant, we have

$$\frac{M_k(l_2;b_2)}{M_k(l_1;b_1)} = \frac{dS(l_1;b_1,b_2)}{dl_1} * \frac{b_1}{b_2} = \frac{M_1(l_2;b_2)}{M_1(l_1;b_1)} \tag{17}$$

'This leads us to the final equation

$$\frac{M_k(l_2;b_2)}{M_1(l_2;b_2)} = \frac{M_k(l_1;b_1)}{M_1(l_1;b_1)} \qquad (18)$$

Since l_2 and l_1 are related by the spatial locality function in the following manner:

$$_{12} = S(l_1;b_1,b_2),$$

and since $M_1(l;b)$ is known by measurement for all b, we can calculate $PI_k(1_2;b_2)$ if we are given $M_k(l_1;b_1)$.

In a later section we present results which demonstrate the accuracy of the predicted curves when compared with the measured curves. This validates the hypothesis that the spatial locality function, as defined by the Stack Working Set Model is invariant in the hierarchy.

We will now model the fault stream out of a level (the request stream to the next level) given the request stream in to the level. The size of the level is n pages of size b bytes. The level is assumed to be fully associative. This is done in Theorem 4. Theorems 2 and 3 are stated since they are necessary for the proof of Theorem 4.

THEOREM 2. The average time taken by a page to drop out of a level of size n pages with a page size of b bytes (using LRU replacement) measured from the instant it last was referenced is given by

$$V(nb;b) = \sum_{i=0}^{n-1} \frac{1}{M(ib;b)}$$

PROOF: This follows quite simply from the Stack Working Set concept. A page is displaced, under LRU replacement, if n distinct pages are referenced from the time it was last referenced. The average time taken to reference n distinct pages is, precisely, the average time taken to build up a working set of n pages. This was earlier defined to be T(nb;b). Therefore,

$$V(nb;b) = T(nb;b) = \sum_{i=0}^{n-1} \frac{1}{M(ib;b)} \qquad (19)$$

A more rigorous proof is given in the Appendix.

THEOREM 3. The average time of residency of a page in a memory of size n pages, measured from the time it is fetched from the lower level to the time that it is displaced from the level in question, is given by

$$W(nb;b) = \frac{n}{M(nb;b)}$$

PROOF: We shall use a Markov chain to derive this result. Let the state of the Markov chain be given by the position of the "marked" page in the LRU stack. Thus, the state is i if the marked page is in position i of the LRU stack. If a page is in position i at time t, at time t+1 it will be in position 1 if it is referenced, position i if a page above it in the stack is referenced and in position i+l if a page below it in the stack is referenced. The probabilities of these three occurences are given by P(ib;b), 1-M((i-1)b;b) and M(ib;b) respectively. The transition graph for this chain is shown in Fig.7 and the transition natrix, B, is given in Fig.8 . By inspection it is clear that the chain is aperiodic and irreducible.

The state n+l corresponds to the page having been displaced from the level. State n+1 is absorbing, for we are interested in examining the average residency of the page for one visit to the level.

Let $x(t;i)$ be the probability that the chain is in state i at time t, where t is measured from the instant that the page is fetched to the level. Therefore, $x(1;1) = 1$. Also, define the probability vector

$$X(t) = [\ x(t;1),\ x(t;2),.....,\ x(t;n+1)\ ] \qquad (20)$$

Therefore,

$$X(1) = [\ 1,0,0,.....,0\ ] \qquad (21)$$

Let $r(t;i)$ be the average time spent in state i over the interval $[1,t]$. Then, define the vector

$$R(t) = [\ r(t;1),\ r(t;2),.....,\ r(t;n+1)\ ]. \qquad (22)$$

Therefore,

$$R(t) = \sum_{j=1}^{t} X(j) \qquad (23)$$

Since $X(t) = X(t-1) * B$, we have (24)

$$R(t) = \sum_{j=1}^{t} X(j) = X(1) + X(1)*B + X(1)*B^2 + \ldots + X(1)*B^{t-1} \quad (25)$$

By letting t tend to infinity we ensure that the process is absorbed by state n+l, i.e. the page is displaced. Let

$R(t) \rightarrow R_{inf}$, and (26)

$X(t) \rightarrow X_{inf}$, as $t \rightarrow$ infinity. (27)

R_{inf} represents the average time spent by the page in each of the n positions in the stack corresponding to the level. We have

$$R_{inf} = X(1) + X(1)*B + \ldots\ldots \quad (28)$$

$$R_{inf}*B = X(1)*B + X(1)*B^2 + \ldots\ldots \quad (29)$$

Subtracting, we get

$$R_{inf}*(I-B) = X(1) - X_{inf} \quad (30)$$

but since state n+l is the only absorbing state,

$$X_{inf} = [\ 0,0,\ldots,0,1\ ]. \tag{31}$$

And since

$$X(1) = [\ 1,0,0,\ldots.,0\ ] \tag{32}$$

we have

$$R_{inf} * (I-B) = [\ 1,0,0,\ldots.,0,-1\ ] \tag{33}$$

The matrix (I-B) is shown in Fig.9. We obtain the following set of equations:

$$M(b;b)*\{\ r_{inf}(2) - r_{inf}(1)\ \} = 0$$
$$M(2b;b)*\{\ r_{inf}(3) - r_{inf}(2)\ \} = 0$$
$$\vdots$$
$$M((n-1)b;b)*\{\ r_{inf}(n) - r_{inf}(n-1)\ \} = 0$$
$$M(nb;b)*r_{inf}(n) = 1$$

The only solution to this set of equations is

$$r_{inf}(i) = 1/M(nb;b) \quad \text{for all } 1<=i<=n. \tag{34}$$

Consequently, the total average residency is given by

$$W(nb;b) = \sum_{i=1}^{n} r_{inf}(i) = \frac{n}{M(nb;b)} \qquad (35)$$

We are now in a position to derive an approximate model for the fault stream out of a level. For the sake of clarity, the notation will be abbreviated by omitting the parameter which specifies the page size. It will be understood that the page size throughout the following analysis is b bytes.

Theorem 4. Under the assumption that all pages have identical behavior, and if $W_k(nb;b)$ and $V_k(nb;b)$ are treated as exact rather than as average times of residency, then if the request stream to level k is characterized by $M_k(l)$, and the size of the level is n pages, then the fault stream is characterized by

$$M_{k+1}(l) = \begin{cases} M_k(nb) & \text{for } 1 < nb \\ M_k(l) & \text{for } 1 >= nb \end{cases}$$

PROOF: Let

$$Q(t) = \frac{1}{b} * \frac{dL(t)}{dt} \quad \text{and} \tag{36}$$

$$F(t) = -\frac{dQ(t)}{dt} \tag{37}$$

F(t) is the the probability that length of the interval between two successive references to any given page is t [DENN72]. Q(t) is the fault probability corresponding to the size of the stack working set built up over a period t.

Note:

$$Q(t) = M(L(t \tag{38}$$

and

$$M(nb) \; \square \; (T(nb)) \tag{39}$$

Let $F_k(t)$ and $Q_k(t)$ correspond to the request stream to level k, and $F_{k+1}(t)$ and $Q_{k+1}(t)$ correspond to the fault stream.

For notational convenience, $W_k(nb;b)$ and $V_k(nb;b)$ will often be referred to as W_k and Vk, without introducing any ambiguity.

A page cannot be referenced at level k+l if a copy is present at level k, for the reference will be intercepted by level k. Therefore, for $t < W_k(nb)$ (as defined in Theor.3),

$$F_{k+1}(t) = 0 \tag{40}$$

If $t >= W_k(nb)$ then the page is no longer present in level k. By Theor. 2, the marked page was referenced $V_k(nb$

time units before the time that it was displaced from level k (Fig.10). Therefore, for $t >= W_k(nb)$,

$$F_{k+1}(t) = F(t - W_k + V_k) \tag{41}$$

Therefore, we have

$$F_{k+1}(t) = 0 \qquad \text{for } t < W_k \tag{42}$$

$$F(t - W_k + V_k) \qquad \text{for } t >= W_k \tag{43}$$

Since, by definition, $F_{k+1}(t) = - dQ_{k+1}(t)/dt$, we have

$$Q_{k+1}(t) = - \int_0^t F_{k+1}(x)\, dx \tag{44}$$

$$= - \int_0^t 0\, dx \qquad \text{for } t < W_k,$$

$$\text{and } Q_{k+1}(W_k) = \int_{W_k}^t F_k(x - W_k + V_k)\, dx \quad \text{for } t>=W_k \tag{45}$$

Now, since every fault in level k is also a fault at level k+l if the working set size at k+l is 0,

$$Q_{k+1}(0) = M_k(nb) \tag{46}$$

But by Eqn.39,

$$M_k(nb) = Q_k(T_k(nb)) \tag{47}$$

And since, by Eqn.19,

$$V_k(nb) = T_k(nb) \tag{48}$$

we have

$$Q_{k+1}(0) = Q_k(V_k) \tag{49}$$

Substituting $\gamma = x - R_k + V_k$ in Eqn.45 and using Eqn.49 we get,

$$Q_{k+1}(t) = Q_k(V_k) \qquad \text{for } t < W_k,$$

$$\text{and } Q_k(V_k) = \int_{V_k}^{t-W_k+V_k} F(y)\, dy \qquad \text{for } t >= W_k. \tag{50}$$

Therefore,

$$Q_{k+1}(t) = \begin{cases} Q_k(V_k) & t < W_k, \\ Q_k(t - W_k + V_k) & t >= W_k. \end{cases} \quad (51)$$

Next, since by definition $Q(t) = 1/b * dL(1)/dt$ we have

$$L_{k+1}(t) = \int_0^t b*Q_{k+1}(x)\, dx \qquad t < W_k \quad (52)$$

$$L_{k+1}(t) = \int_0^t b*Q_k(V_k)\, dx \qquad t < W_k,$$

$$\text{and } L_{k+1}(W_k) + \int_{W_k}^t b*Q_k(x - W_k + V_k)\, dx \qquad t >= W_k \quad (53)$$

Therefore, since $L_{k+1}(0) = 0$,

$$L_{k+1}(t) = b*Q_k(V_k)*t \qquad t < W_k,$$

$$b*Q_k(V_k)*W_k + L_k(t - W_k + V_k) - L_k(V_k) \quad t >= W_k \quad (54)$$

But since by Eqns.19,3,35,39,

$$L_k(V_k) = L_k(T_k(nb)) \cdot nb, \text{ and} \tag{55}$$

$$W_k(nb) = n/M_k(nb) \text{ and} \tag{56}$$

$$Q_k(V_k) = Q_k(T_k(nb)) = M_k(nb), \tag{57}$$

we have,

$$L_{k+1}(t) = b*M_k(nb)*t \qquad t < W_k$$

$$L_k(t - W_k + V_k) \qquad t >= W_k. \tag{58}$$

Therefore, for $t < W_k$ we have

$$L_{k+1}(t) <= nb, \text{ and} \tag{59}$$

and by Eqn .38

$$M_{k+1}(L_{k+1}(t) = Q_{k+1}(t) \tag{60}$$

by Eqn.51,

$$Q_{k+1}(t) = Q_k(V_k), \tag{61}$$

by Eqn.19,

$$Q_k(V_k) = Q_k(T_k(nb)), \tag{62}$$

and by Eqn.39,

$$Q_k(T_k(nb)) = M_k(nb) \tag{63}$$

Therefore,

$$M_{k+1}(L_{k+1}(t)) = M_k(nb) \tag{64}$$

$$\text{i.e. } M_{k+1}(1) = M_k(nb) \qquad \text{for } 1 < nb \tag{65}$$

For $t >= W_k$ we have

$$L_{k+1}(t) >= nb, \text{ and} \tag{66}$$

by Eqn.38,

$$Q_{k+1}(t) = M_{k+1}(L_{k+1}(t)) \tag{67}$$

Since by Eqn.51,

$$Q_{k+1}(t) = Q_k(t - W_k + V_k) \text{ and} \tag{68}$$

and by Eqn.58,

$$L_{k+1}(t) = L_k(t - W_k + V_k) \tag{69}$$

Therefore, by Eqn.67,

$$Q_k(t - W_k + V_k) = M_{k+1}(L_k(t - W_k + V_k)) \tag{70}$$

But, by Eqn.38,

$$Q_k(t - W_k + V_k) = M_k(L_k(t - W_k + V_k)) \qquad (71)$$

Therefore, from Eqns.69,70,

$$M_{k+1}(1) = M_k(1) \qquad \text{for } 1 >= nb \qquad (72)$$

We now have the theorem in its final form:

$$M_{k+1}(1) = \begin{cases} M_k(nb & 1 < nb, \\ M_k(1) & 1 >= nb. \end{cases} \qquad (73)$$

Theorem 4 states that, for a constant page size, the top n paces in the LRU stack for level k+l will be in the higher level (of size n pages). Conversely, it also states that a page which is at a position lower than n in the LRU stack for level k+l will not be in level k. This is, admittedly, a rather simplistic analysis since it uses only the average residency times and does not take into account the variance. In a the next section we shall see that it is, nevertheless, fairly accurate for small page sizes.

V. EXPERIMENTAL VERIFICATION

All experiments were conducted by simulating a stream of requests to a memory hierarchy. The stream was, in each case, obtained from one of three adress traces which were created by an instruction-by-instruction trace of actually executing programs. The trace tapes used are:

DRC050 - Cobol compilation
DRC043 - Fortran execution
DRC049 - Cobol execution

In Figs.11,12,13 we verify the validity of the hypothesis that the spatial locality of a request stream is invariant. We do so by using Theorem 3 to analytically arrive at the miss rate curve, $M_2(l;b_1,b_2)$, at level 2. The predicted curve is then compared with the measured one. The highest level was assumed to be of size 2048 bytes and employing a page (or block) size of 32 bytes. The second level was assumed to have a page size of 128 bytes. The results obtained indicate that it is reasonable to consider the spatial locality to be an invariant of the hierarchy. Fig.14 presents the spatial locality function, $S(l;32,128)$, for the trace tape DRC050, which was used to derive the function $M_2(l;128)$. It is interesting to note that for small values of l_1, the ratio l_2/l, is very large, but that

it tends to 1 as l_1 increases. This is in agreement with the intuitive notion that the wastage in going to a larger page size is less if the larger pages are kept around long enough to have all their sub-pages referenced.

In Fig.15 we test the invariance of the spatial locality for larger page sizes. DRC050 is analyzed using a memory at level 1 of size 8K bytes, page size 1K bytes and a page size of 21; bytes at level 2. Once again, the agreement between the predicted and measured curves is satisfactory.

Figs.16 and 17 are comparisons of the measured fault stream out of level 1 and the predicted fault stream obtained using Theorem 4. Fig.16 is for DRC043, and represents the worst results obtained. Fig.17, for DRC049, represents the best results. The parameters of the simulation were the same as those for Figs.11,12,13. Fig.18 is a comparison of the measured and the predicted fault stream for DRC050 for a larger page size. The page size for level 1 is 1K bytes, the level being of size 8K bytes. The page size for the second level is 2K bytes. It can be seen that the predicted curve deviates quite a lot from the measured one for memory sizes of about 8K bytes (the size of the higher level). It is clear that the assumptions on which Theorem 4 are based become increasingly invalid as the page size is increased. The model for the fault stream can

be improved by taking into account the variance in the residency time of a page in a level. However, it is felt that another factor exists which contributes to the inaccuracy of the model. Each page behaves differently and, therefore, different pages will have different distributions for their time of residency. The overall averaged behavior taking this into account will show a variance which need not be that of a normal distribution as would be predicted by the SWSM model. This is a fundamental drawback of the LRUSM, and, hence, the SWSM, in that it assumes that all pages are statistically identical.

We can use the two-level miss rate statistics gathered for a request stream to predict the fault rate at any level in the hierarchy. We first use Theorem 4 to model the fault stream out of level 1 assuming a page size of b_1. We then use the miss rate statistics measured at level 1 for block size b_2 and, using the invariance of the spatial locality, derive a characterization for the fault stream with a block size of b_2. We can, iteratively, descend into the hierarchy, and compute the miss rates at each level. The weak link in this process is the characterization of the fault stream, and the accuracy of the performance evaluation of a hierarchy is limited by the adherence of the request stream to the assumptions underlying Theorem 4. Fig.19 displays the predicted miss rate out of level 2 obtained in

this way, for a page size at level 1 of 32 bytes, a memory of 2048 bytes at level 1 and a page size of 128 bytes at level 2 for DRC050.

VI. EXTENSION OF THE CHARACTERIZATION OF SPATIAL LOCALITY TO DIFFERENT MODELS

The spatial locality function $S(1;b_1,b_2)$ can be incorporated into any model which is capable of predicting the working set size accurately as a function of time. That the SWSM is one such model is evidenced by Fip.20 which compares the WSM and the SWSM. In addition to the WSM and the SWSM, the AO-inversion Independent Reference Model appears to be another model which is capable of predicting the working set size accurately [BASK75]. If it is possible to make transformations in both directions, from miss rate statistics to working set statistics, and vice versa, techniques very similar to those outlined above can be used. The WSM should be at least as effective as the SWSM in characterizing the fault stream. The derivation for this is included in Theorem 4. It is not quite as clear whether the AO-inversion model will be as tractable in this respect.

The close agreement between the predicted average working set size for the WSM and the SWSM (Fig.20) is reason to believe that the SWSM might be a good estimator of performance when using variable memory allocation. This is significant, for the SWSM derives its statistics from the LRUSM, which requires the gathering of far fewer statistics than does the WSM.

VII. THE RELATIONSHIP OF THIS WORK TO PREVIOUS WORK

The use of stack processing was first proposed by Mattson, et al,[MATT70] and constitutes the backbone for the LRUSEI. They indicated the applicability of this technique to a type of multilevel hierarchy in which the page size is constant throughout. Furthermore, multiple copies of pages at different levels were not permitted in their model of the hierarchy.

Slutz and Traiger [SLUT72] improved on these restrictive assumptions and allowed the use of different page sizes at different levels and also allowed for the fact that each unit of information could have a copy at a nunber of levels. However, they required that every request from the processor be broadcast to every level, so that each level could order its pages just as if it were the highest level. They then showed that, if each level has at least as many pages (of equal or larger size) as the level above, then the miss rate out of a level would be independent of the levels above it. Unfortunately, it is not practical to broadcast requests to every level since the lower levels, being designed with slower technology will not be able to handle the request rate to the highest level.

In this work, each level sees only the fault stream out of the level immediately above it, and no restrictions are placed on the relative sizes of the levels. However, it is assumed that an LRU policy is used at every level and that a store-through policy is employed. (However, pages may be moved up either for read faults only, or for both write, as well as read, faults. The stack measurements must be made accordingly). This last assumption is neccessary to characterize the fault stream in the simple fashion outlined in Theorem 4. If a store-through policy is not enforced, a page which is replaced might need to be written back to the parent page in the lower level which contains it. This constitutes additional requests which cannot be handled by the SWSM in its present form.

VIII. CONCLUSION

We have characterized the spatial locality of a request stream such that it is independent of the temporal locality, and have shown that it is an invariant in the hierarchy.

We have characterized the temporal locality of a request stream in terms of the smallest page size possible so that it is independent of the spatial locality. We have indicated an approximate technique for deriving the temporal locality of the request stream for any page size at any level in the hierarchy. Since the miss rate under the LRU replacement policy is closely related to the characterization of temporal locality, we are able to evaluate the performance of a general multilevel hierarchy.

We have shown that the LRU stack statistics gathered for all the page sizes of interest contain all the information necessary to characterize the spatial and temporal locality of a request stream and to evaluate the performance of a multilevel hierarchy. These statistics can all be gathered in one pass of the address trace [MATT70].

The Stack Working Set model, based on these statistics, satisfies the first three requirements of the "ideal" model for multilevel hierarchies, as outlined in Section.III, quite well.

BIBLIOGRAPHY

AHO72 Aho,A.V., P.J.Denning and J.Ullman, "Principles of Optimal Page Replacement", JACM 18, 1, (Jan 1971)

ANAC67 Anacker,W., and C.P.Wang, "Performance Evaluation Of Computer systems With Memory Hierarchies", IEEE-TEC EC-16, 6, (Dec 1967)

BASK75 Baskett,F., and A.Rafii, "The AO Inversion Model Of Program Paging Behavior", Stanford Univ., Comp. Sci. Dept., Tech. Rep. (to be published)

BELA66 Belady,L.A., "A Study Of Replacement Algorithms For A Virtual-Storage Computer", IBM Sys. Jour., 5, 2, (1966)

BELA69 Belady,L.A., and C.J.Keuhner, "Dynamic Space Sharing In Computer Systems", CACM 12, 5, (May 1969)

COFF72 Coffman, E.G., and T.A.Ryan, "A Study Of Storage Partitioning Using A Mathematical Model Of Locality", CACM 15, 3, (Mar 1972)

COFF73 Coffman, E.G., and P.J.Denning, "Operating Systems Theory", Prentice-Hall (1973), Pgs. 241-312

DENN68 Denning,P.J., "The Working Set Model For Program Behavior", CACM 11, 5, (May 1968)

DENN70 Denning,P.J., "Virtual Memory", Comp. Surveys, 2,3, (Sep 1970)

DENIJ72 Denning,P.J., and S.C.Schwartz, "Properties Of The Working Set Model", CACM 11, 5, (Mar 1972)

FRAN74 Franaszek,P.A., and T.J.Wagner, "Some Distribution-Free Aspects Of Paging Algorithm Performance"JACM 21, 1, (Jan 1974)

GECS74 Gecsei,J., "Determining Hit Ratios for Multilevel Hierarchies", IBM Jour. Res. Dev. (July 1974)

HATF72 Hatfield,D.J., "Experiments on Pace Size, Program Access Patterns, and Virtual Memory Performance", IBM Jour. Res. Dev. (Jan 1972)

KING71 King,W.F., "Analysis of Paging Algorithms", IBM T.J.Watson Res. Cen. Rep., RC3288 (Mar 1971)

MADN72 Madnick,S.E., "Storage Hierarchy Systems", MIT, Elec. Engg. Dept., Ph.D. Thesis

MATT70 Mattson,R.L., J.Gecsei, D.R.Slutz and I.L.Traiger, "Evaluation Techniques for Storage Hierarchies**, IBM sys. Jour., 2, (1970)

RAO75 Rao,G.S., **Performance Evaluation of Cache Memories", Stanford Univ., Dept. Elec. Engg., Ph.D. Thesis

SALT74 Saltzer,J.H., "A Simple Linear Model of Demand Paging Performance**, CACM 17, 4, (Apr 1974)

SHED72 Shedler,G.S., and C.Tung, "Locality in Page Reference Strings**, SIAM J. Comput. 1, 3, (Sep 1972)

SHEM66 Shemer,J.E., and G.A.Shippey, **Statistical Analysis of Paced and Segmented Computer Systems**, IEEE-TEC, EC-15, 6, (Dec 1966)

SLUT72 Slutz,D.R., and I.L.Traiger, **Determination of Hit Ratios for a Class of Staging Hierarchies**, IBM San Jose Res. Lab. RJ 1044, (May 1972)

SLUT74 Slutz,D.R., and I.L.Traiger, "A Note on the Calculation of Average Working Set Size**, CACM 17, IO, (Oct 1974)

APPENDIX

A more rigorous proof of Theorem 2 is given here. The average time that a page spends in a level, measured from the instant that it was last referenced to the time that it drops out of the level is given by

$$V(nb;b) = \sum_{i=1}^{n} U(i;b)$$

where $U(i;b)$ is the average time spent by a page in position i of the level given that the next position it goes to is $i+1$ (and not 1). This condition is necessary since the page is not referenced in the interval from the instant of last reference to the instant it drops out of the level, i.e., reaches position n+l.

As a result of this condition, $U(i;b)$ is given by the average length of a realization of the form

$$s_2 = i,\ s_3 = i,\ldots,\ s_j = i,\ s_{j+1} = i+1 | s_1 = i$$

where s_j is the position of the page in the stack, and where $j = 2, 3, \ldots$ infinity.

Therefore,

$$U(i;b) = \sum_{j=1}^{inf} j*P[s_2=i,\ldots,s_j=i,s_{j+1}=i+1 | s_1=i \text{ and the next distinct position visited is } i+1]$$

Now,

$$P[s_2=i,\ldots,s_j=i,s_{j+1}=i+1 | s_1=i] = M(ib;b)*[1-M((i-1)b;b)]^{j-1}$$

and,

$$P[\text{next distinct position visited is } i+1 | s_1=i]$$

$$= \sum_{j=1}^{inf} P[s_2=i,\ldots s_j=i,s_{j+1}=i+1 | s_1=i]$$

$$= \sum_{j=1}^{inf} M(ib;b)*[1-M((i-1)b;b)]^{j-1}$$

$$= M(ib;b)/M((i-1)b;b)$$

Therefore,

$$P[s_2=i,\ldots,s_j=i,s_{j+1}=i+1 \mid s_1=i \text{ and the next distinct position visited is } i+1]$$

$$= M(ib;b)*[1-M((i-1)b;b)]^{j-1}/[M(ib;b)/M((i-1)b;b)]$$

$$= M((i-1)b;b)*[1-M((i-1)b;b)]^{j-1}$$

Therefore,

$$\hat{U}(i;b) = \sum_{j=1}^{\inf} j*M((i-1)b;b)*[1-M((i-1)b;b)]^{j-1}$$

$$= 1/M((i-1)b;b)$$

Therefore,

$$V(nb;b) = \sum_{i=1}^{n} \frac{1}{M((i-1)b;b)} = \sum_{i=0}^{n-1} \frac{1}{M(ib;b)}$$

The author would like to acknowledge Frank Yu for his critical comments which led to this Appendix.

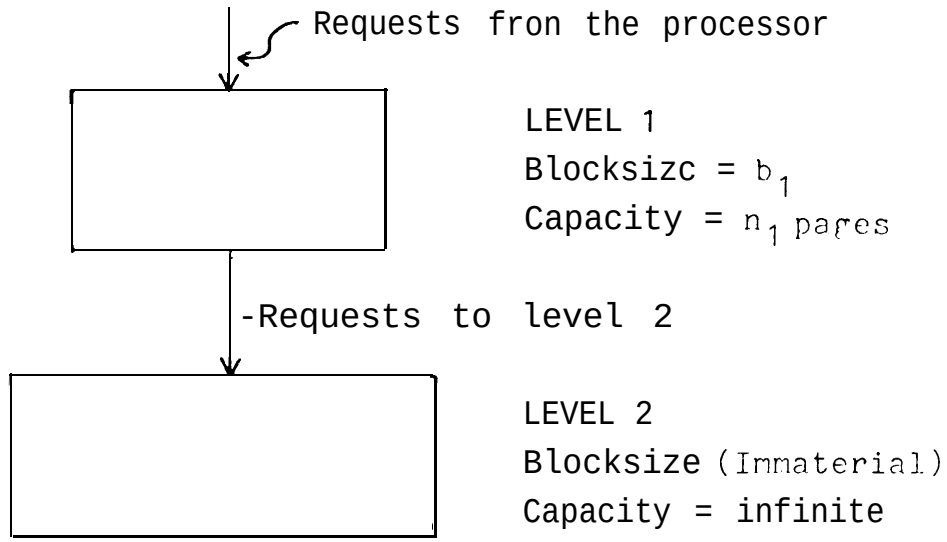
Requests fron the processor
LEVEL 1
Blocksizc = b_1
Capacity = n_1 pages
-Requests to level 2
LEVEL 2
Blocksize (Immaterial)
Capacity = infinite

FIG 1

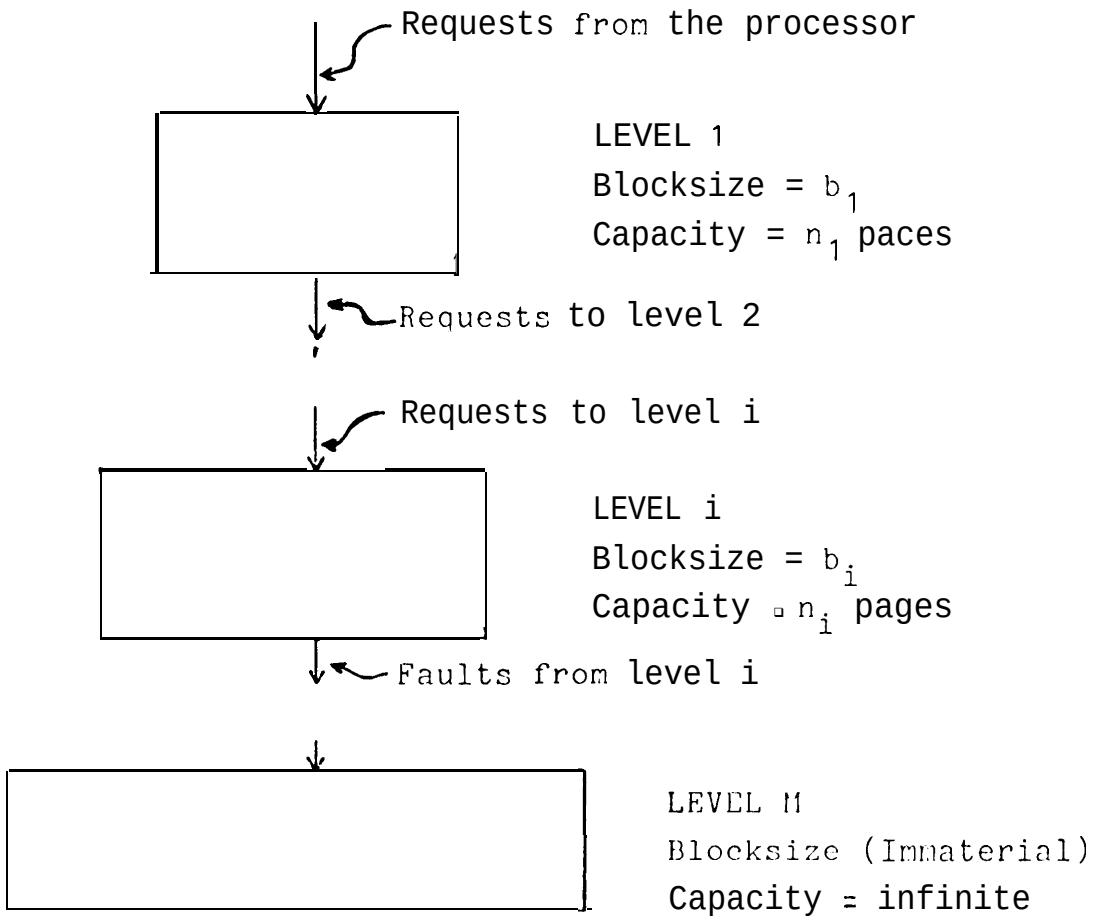
Requests from the processor
LEVEL 1
Blocksize = b_1
Capacity = n_1 paces
Requests to level 2
Requests to level i
LEVEL i
Blocksize = b_i
Capacity = n_i pages
Faults from level i
LEVEL M
Blocksize (Immaterial)
Capacity = infinite

FIG 3

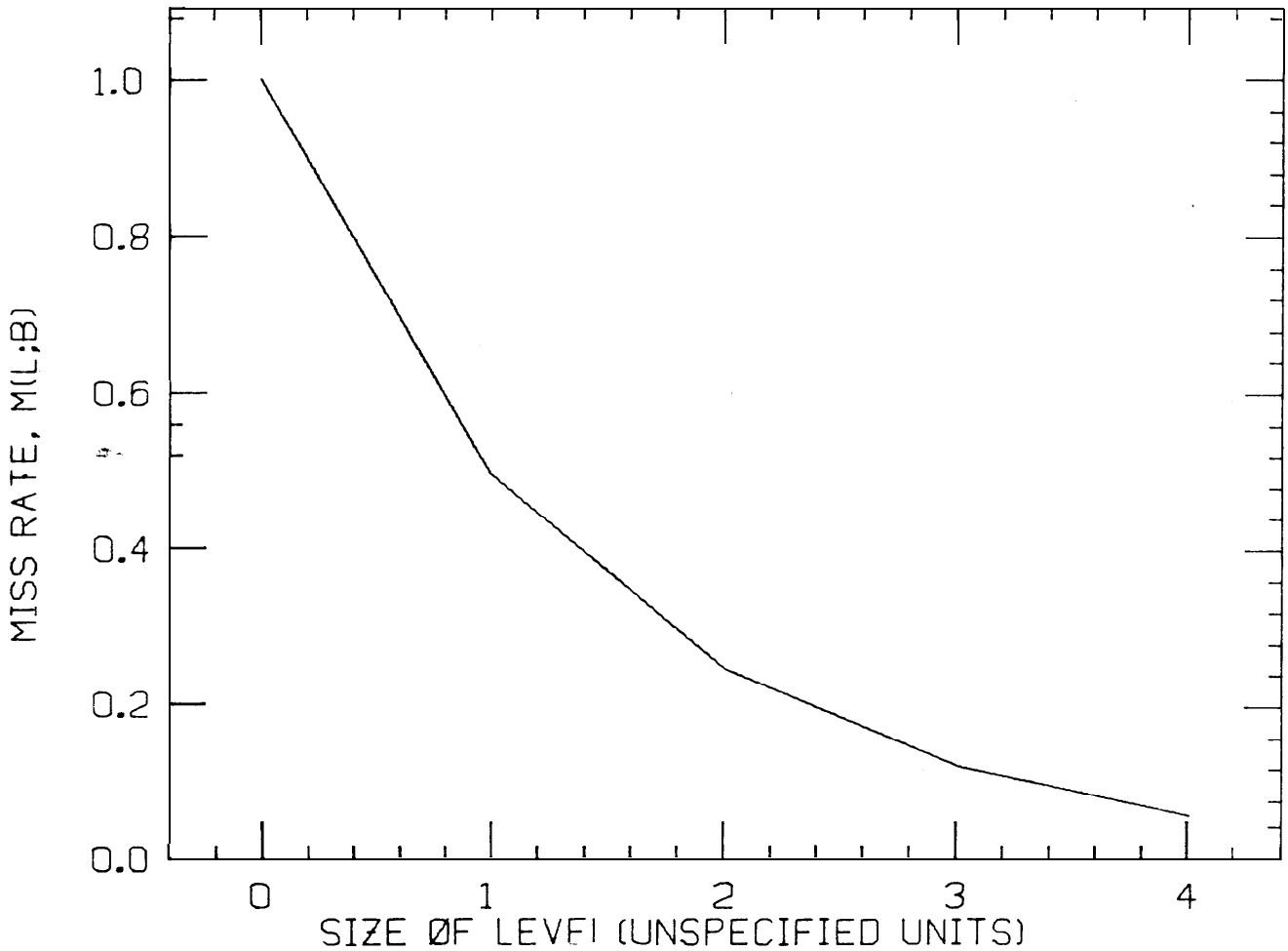
1.0
0.8
0.6
0.4
0.2
0.0
MISS RATE, M(L;B)
0
1
2
3
4
SIZE ØF LEVEI (UNSPECIFIED UNITS)

FIG 3

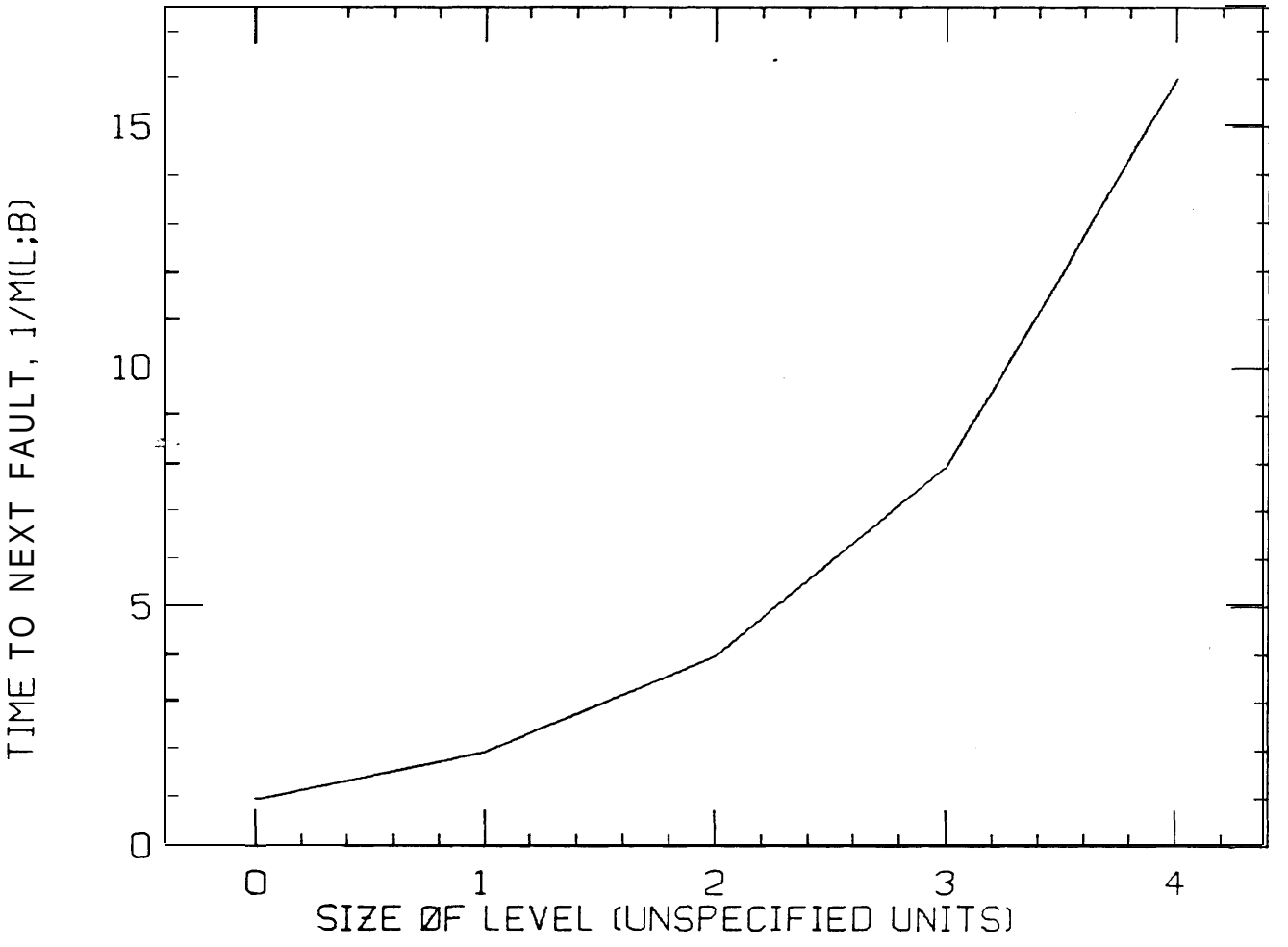
TIME TO NEXT FAULT, 1/M(L;B)
0
5
10
15
0
1
2
3
4
SIZE ØF LEVEL (UNSPECIFIED UNITS)

FIG 4

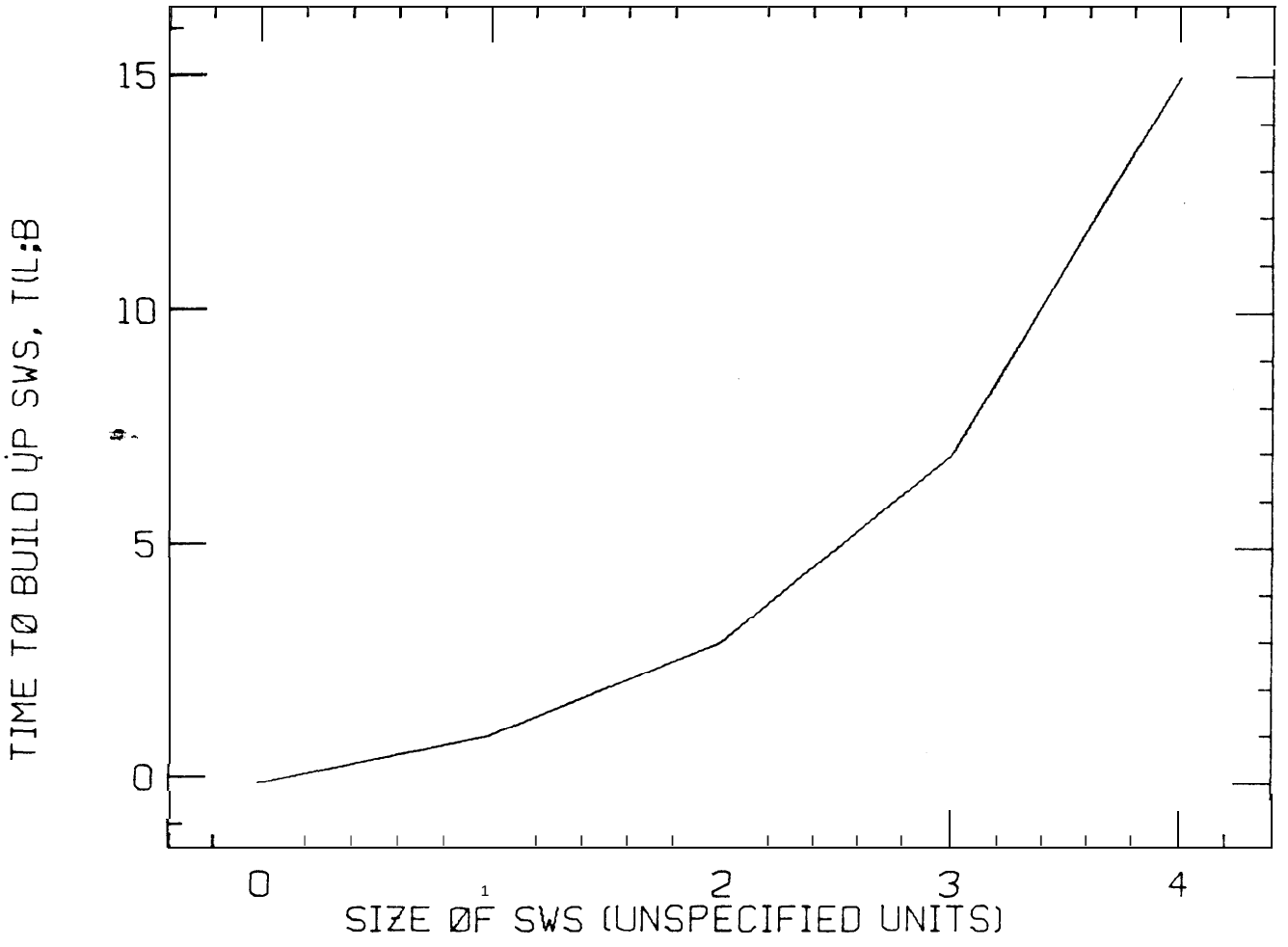
15
10
5
0
TIME TØ BUILD ÚP SWS, T(L;B
0
1
2
3
4
SIZE ØF SWS (UNSPECIFIED UNITS)

FIG 5

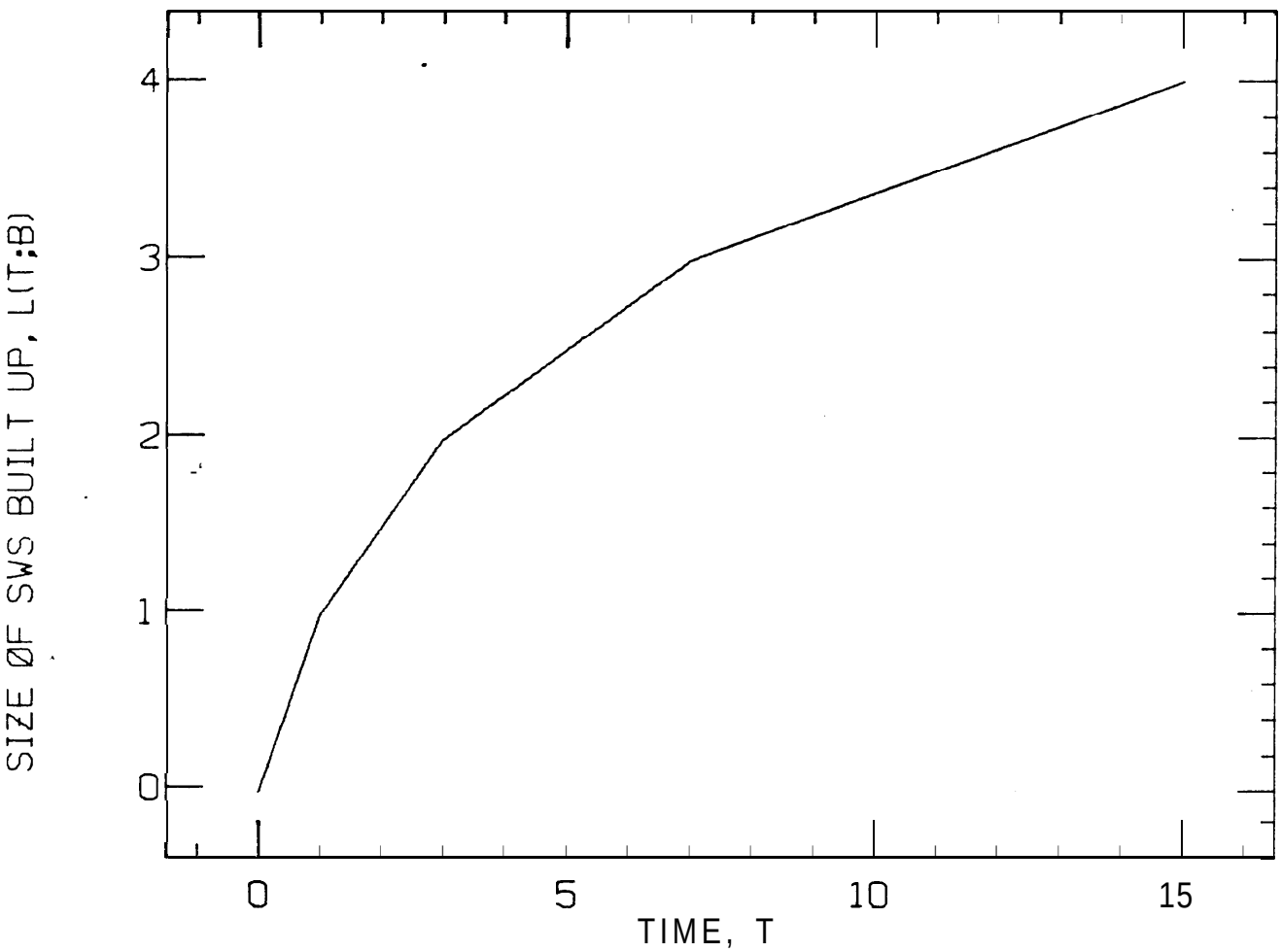

FIG 6

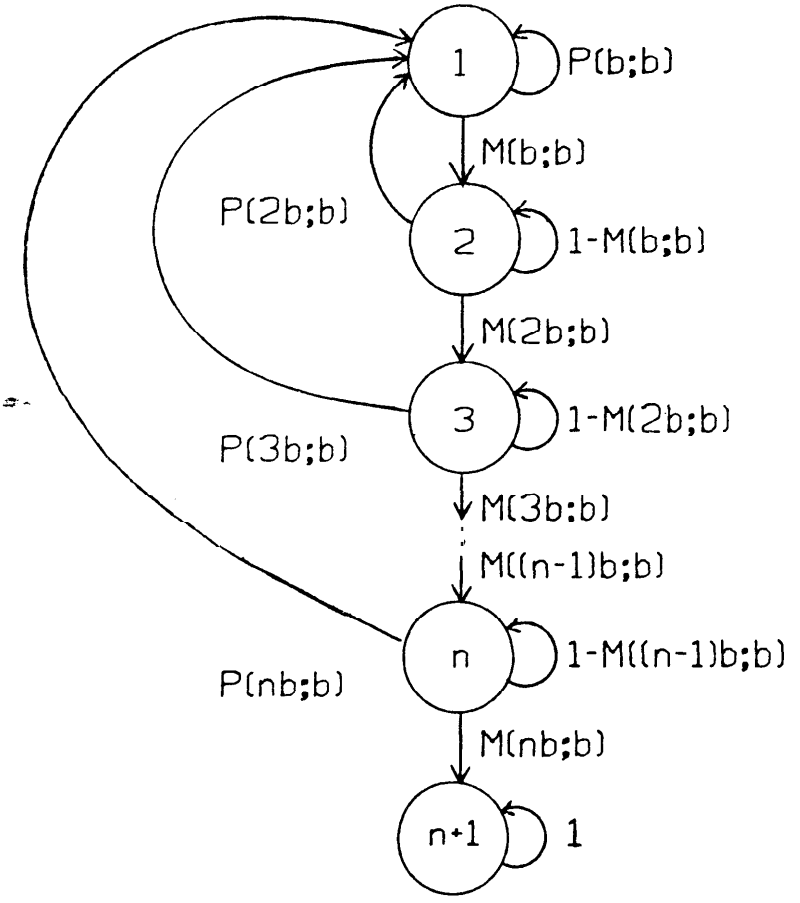
1
P(b;b)
M(b;b)
P(2b;b)
2
1-M(b;b)
M(2b;b)
3
1-M(2b;b)
P(3b;b)
M(3b:b)
M((n-1)b;b)
n
1-M((n-1)b;b)
P(nb;b)
M(nb;b)
n+1
1

FIG 7

		2		n	n+1
1	P(n;b)	H(b;b).....		0	0
2	P(2n;b)	1-H(b;b).....		0	0
3	P(3n;b)	0	.....	0	0
.	.	.	.....	.	.
n	P(nb;b)	0	.....	1-H((n-1)b;b)	H(nb;b)
n+1	0	0	.....	.	1

FIG 8

	1	2	.....	n	n+l
1	M(b;b)	-M(b;b)	.....	0	0
2	-P(2b;b)	M(b;b)	.....	0	0
3	-P(3b;b)	0	.....	0	0
	.	.	.....	.	.
n	-P(nb;b)	0	.....	M((n-1)b;b)	-M(nb;b)
n+l	0	0	.....	0	0

FIG 9

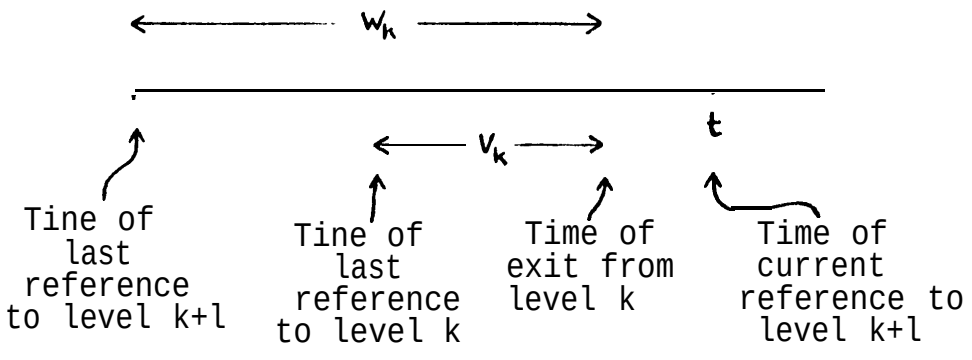

FIG 10

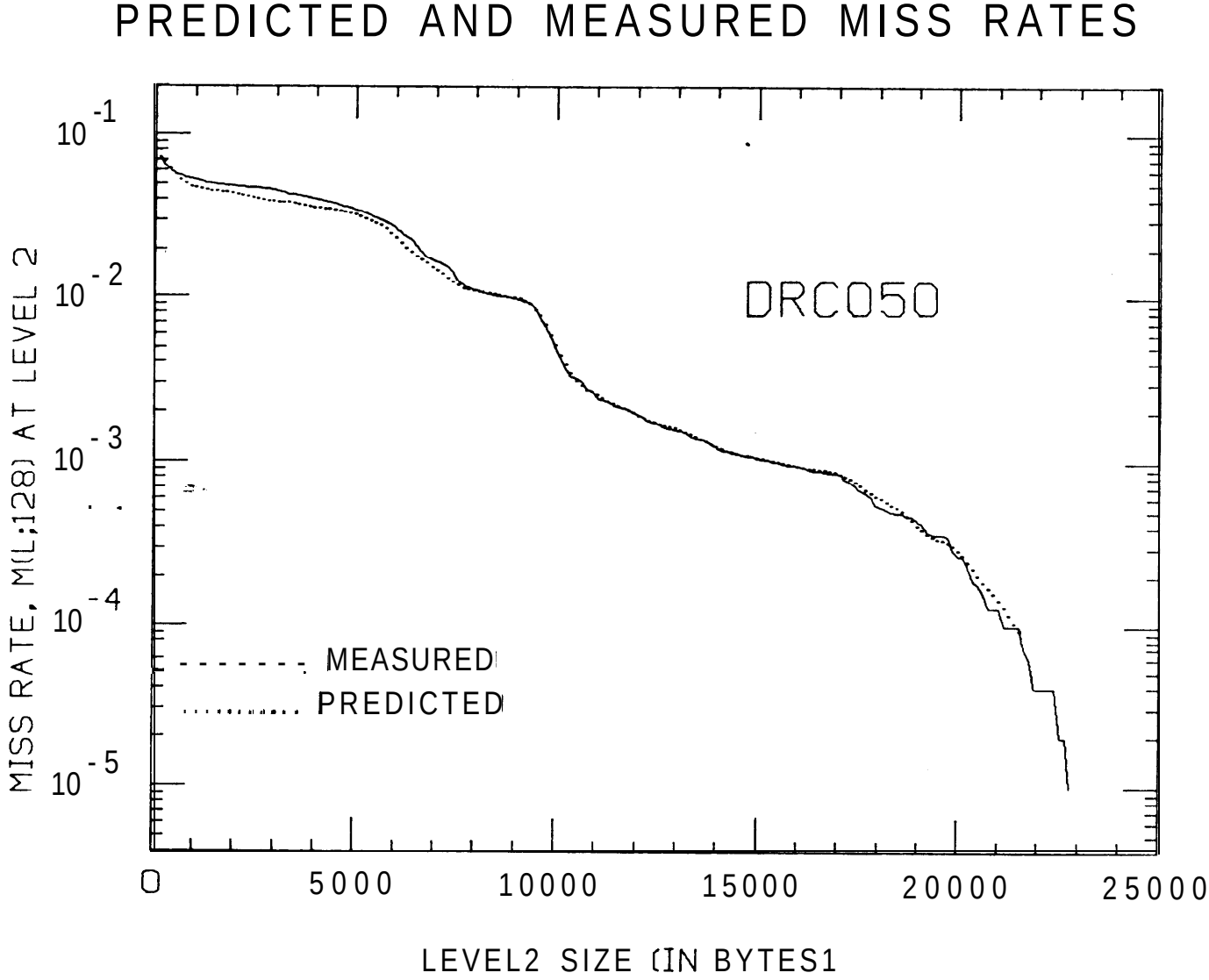
PREDICTED AND MEASURED MISS RATES
MISS RATE, M(L;128) AT LEVEL 2
10 -1
10 -2
10 -3
10 -4
10 -5
DRC050
MEASURED
PREDICTED
0
5000
10000
15000
20000
25000
LEVEL2 SIZE (IN BYTES1

FIG 11

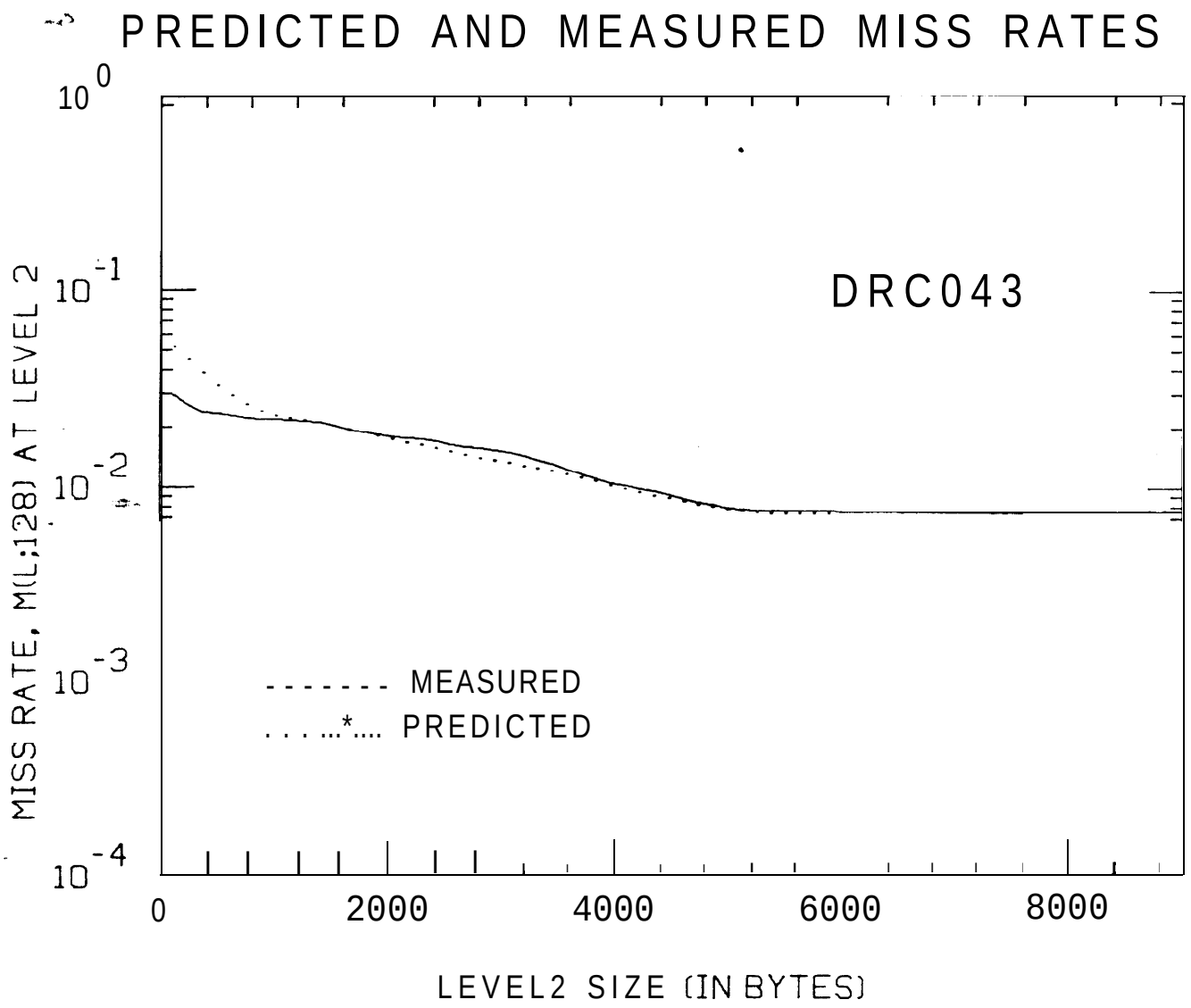
PREDICTED AND MEASURED MISS RATES
DRC043
MISS RATE, M(L;128) AT LEVEL 2
LEVEL2 SIZE (IN BYTES)
MEASURED
PREDICTED
10^0
10^-1
10^-2
10^-3
10^-4
0
2000
4000
6000
8000

FIG- 12

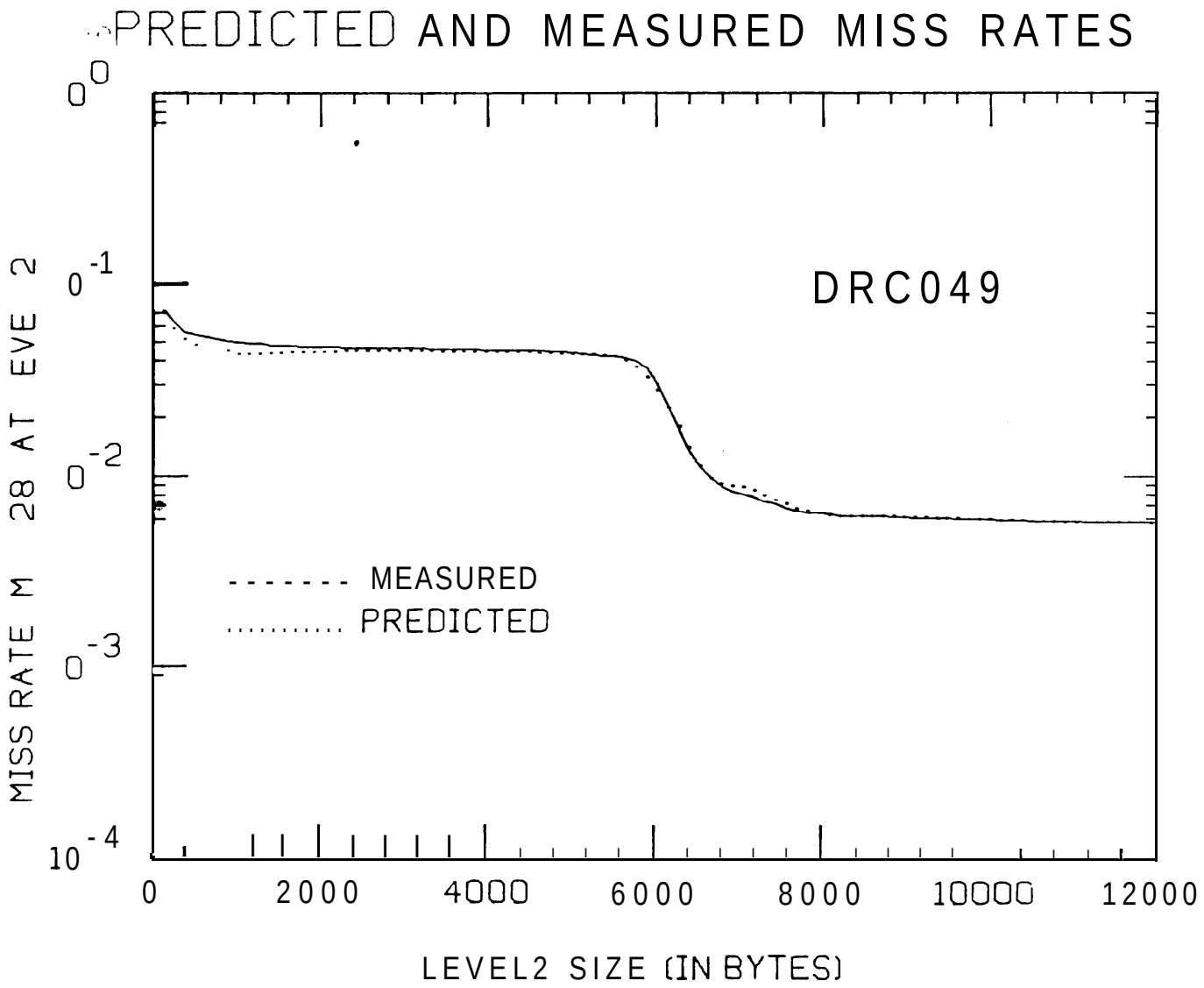
PREDICTED AND MEASURED MISS RATES
DRC049
MEASURED
PREDICTED
MISS RATE M 28 AT EVE 2
LEVEL2 SIZE (IN BYTES)
0
2000
4000
6000
8000
10000
12000

FIG- 13

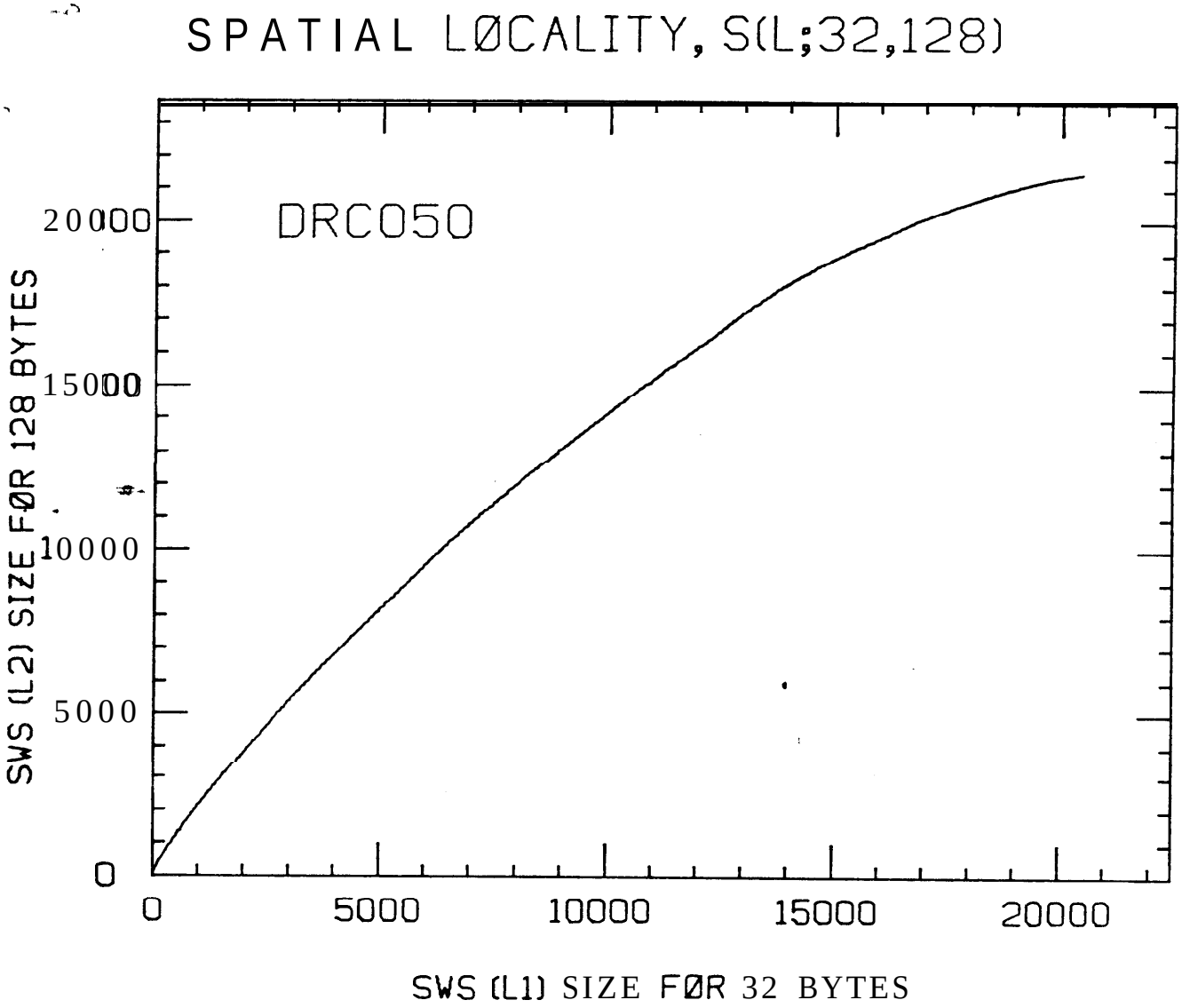
SPATIAL LØCALITY, S(L;32,128)
DRC050
SWS (L2) SIZE FØR 128 BYTES
20000
15000
10000
5000
0
0
5000
10000
15000
20000
SWS (L1) SIZE FØR 32 BYTES

FIG 14

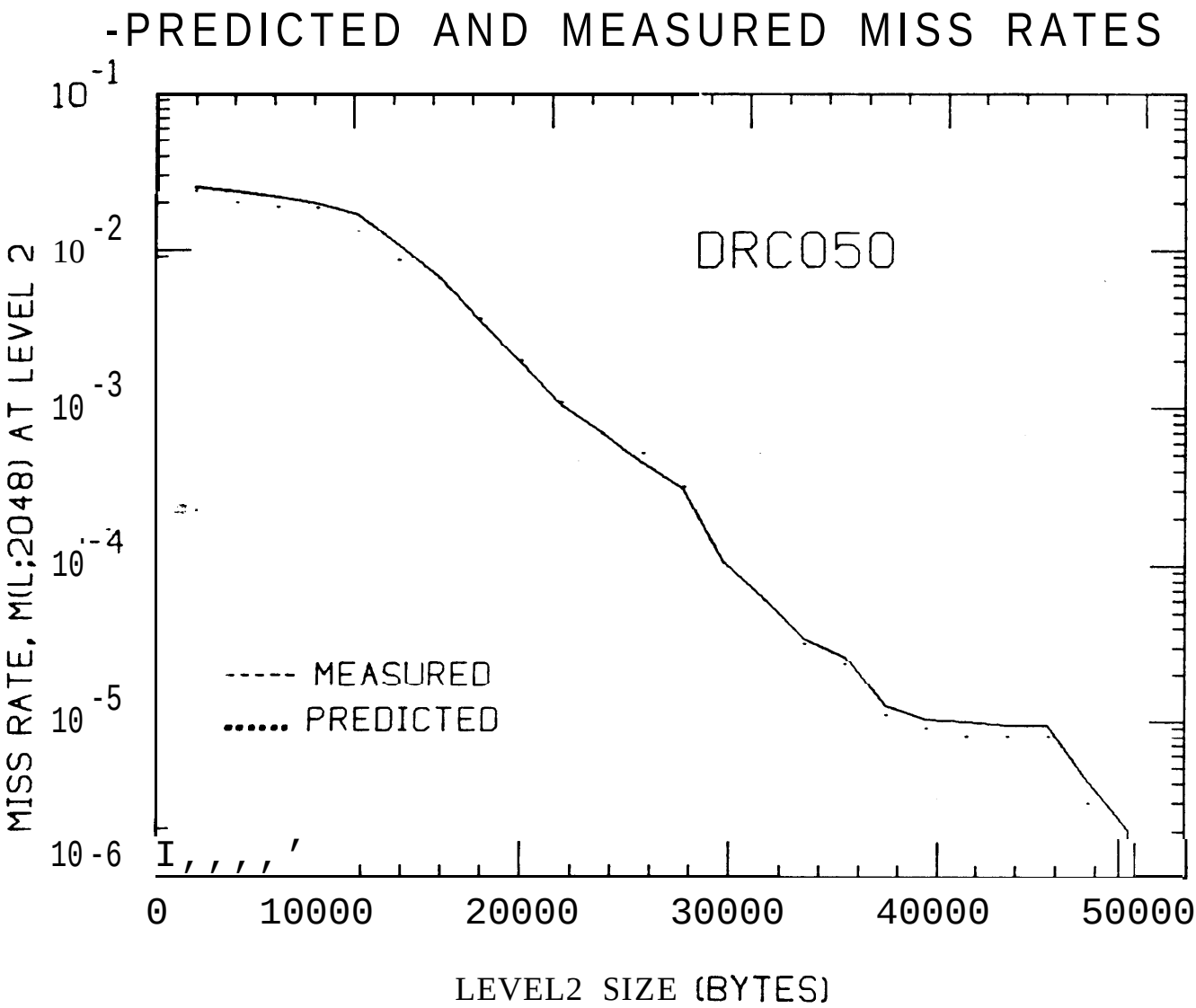
-PREDICTED AND MEASURED MISS RATES
DRC050
----- MEASURED
..... PREDICTED
MISS RATE, M(L;2048) AT LEVEL 2
10-1
10-2
10-3
10-4
10-5
10-6
0
10000
20000
30000
40000
50000
LEVEL2 SIZE (BYTES)

FIG 15

-PREDICTED AND MEASURED FAULT STREAMS

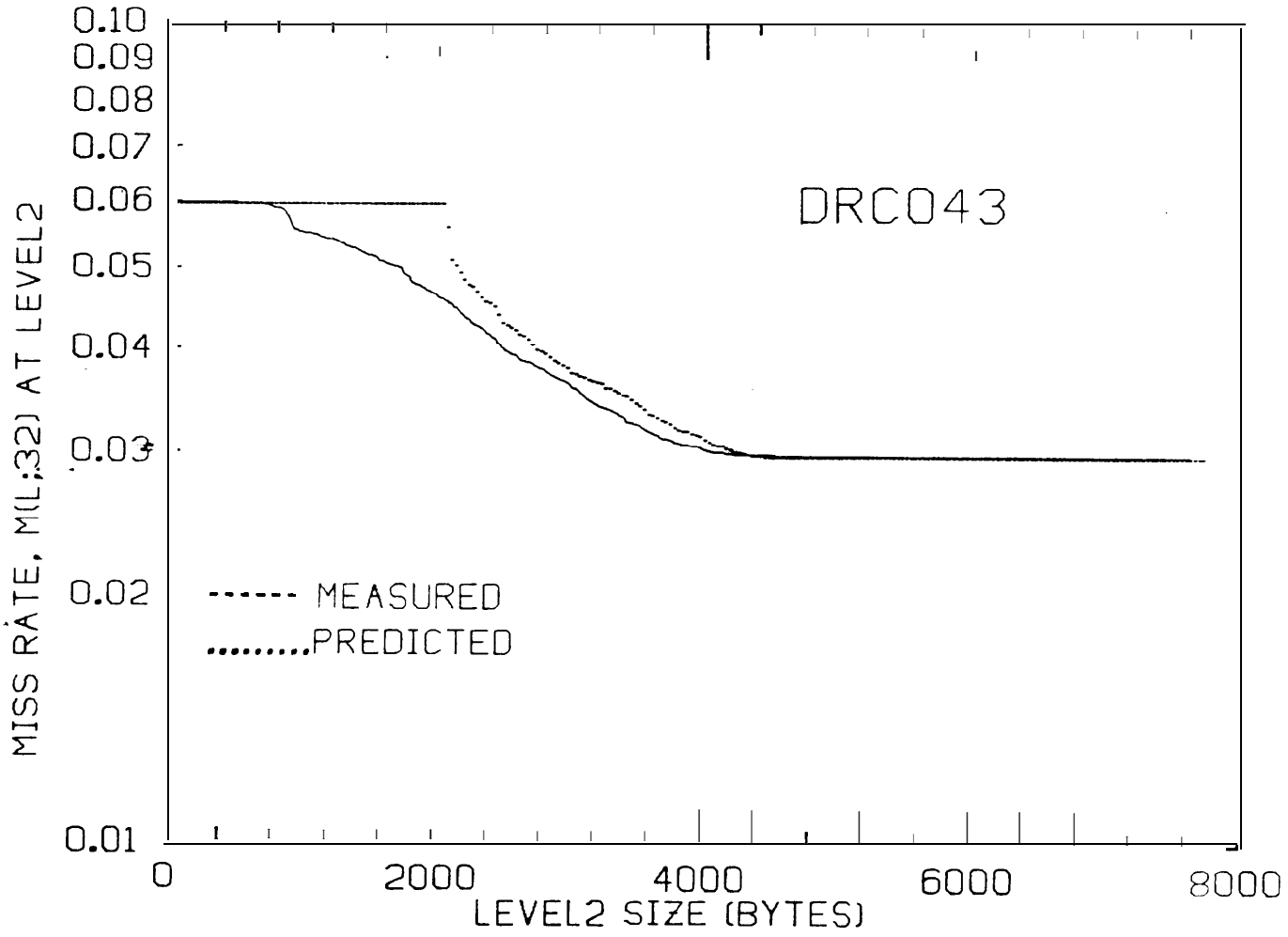

FIG 16

PREDICTED AND MEASURED FAULT STREAMS

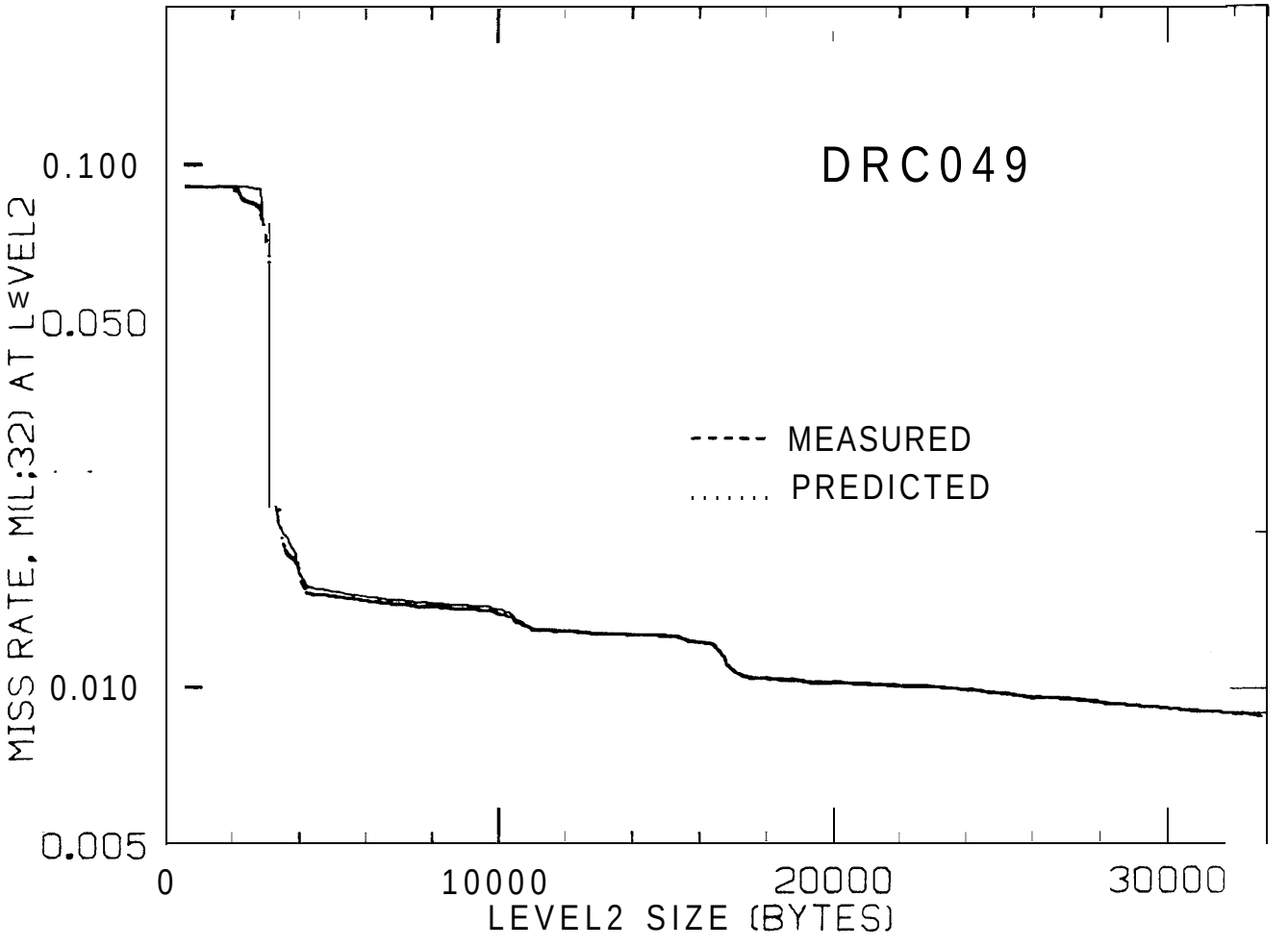

FIG 17

-PREDICTED AND MEASURED FAULT STREAM5

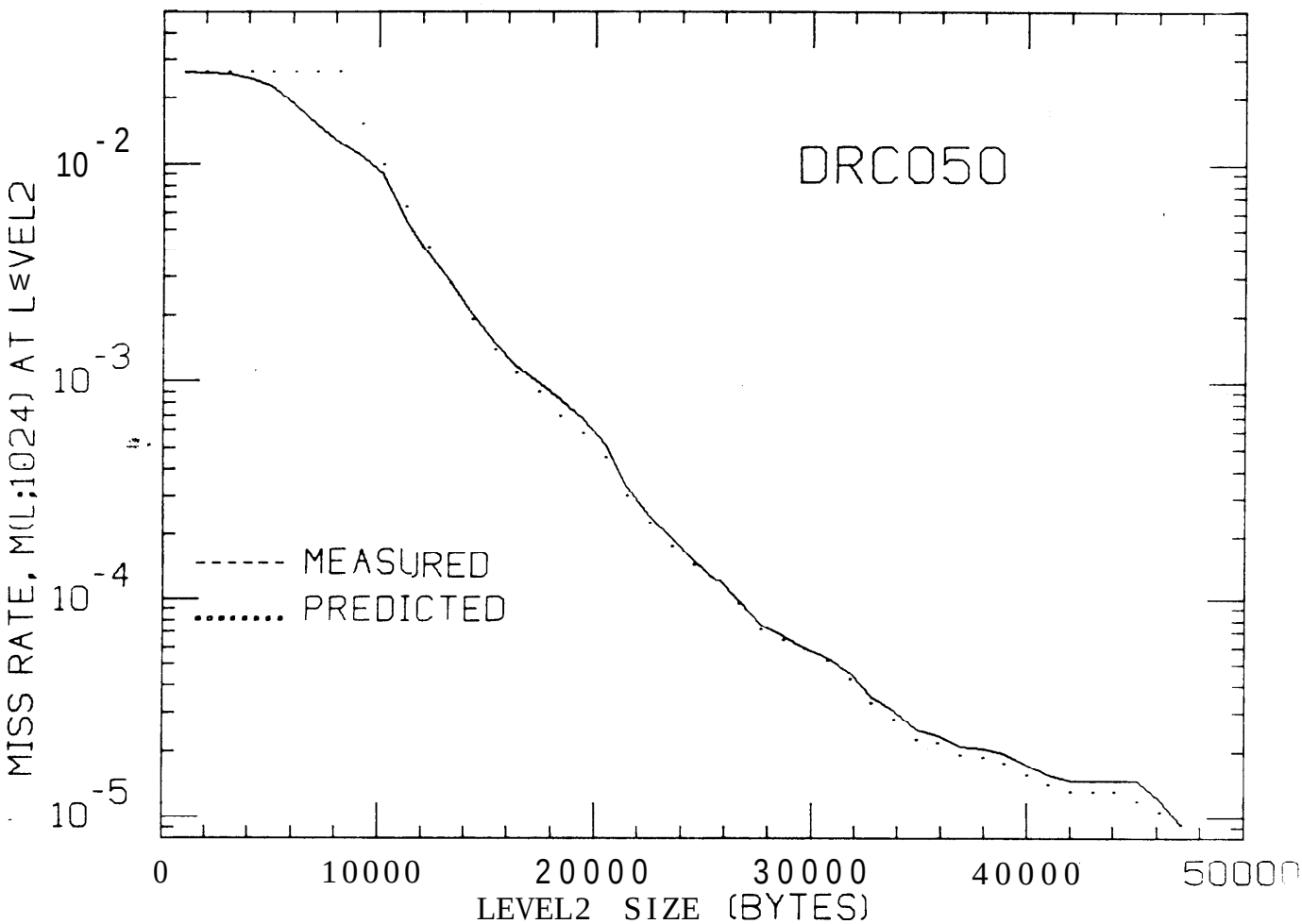

FIG 18

PREDICTED AND MEASURED MISS RATES

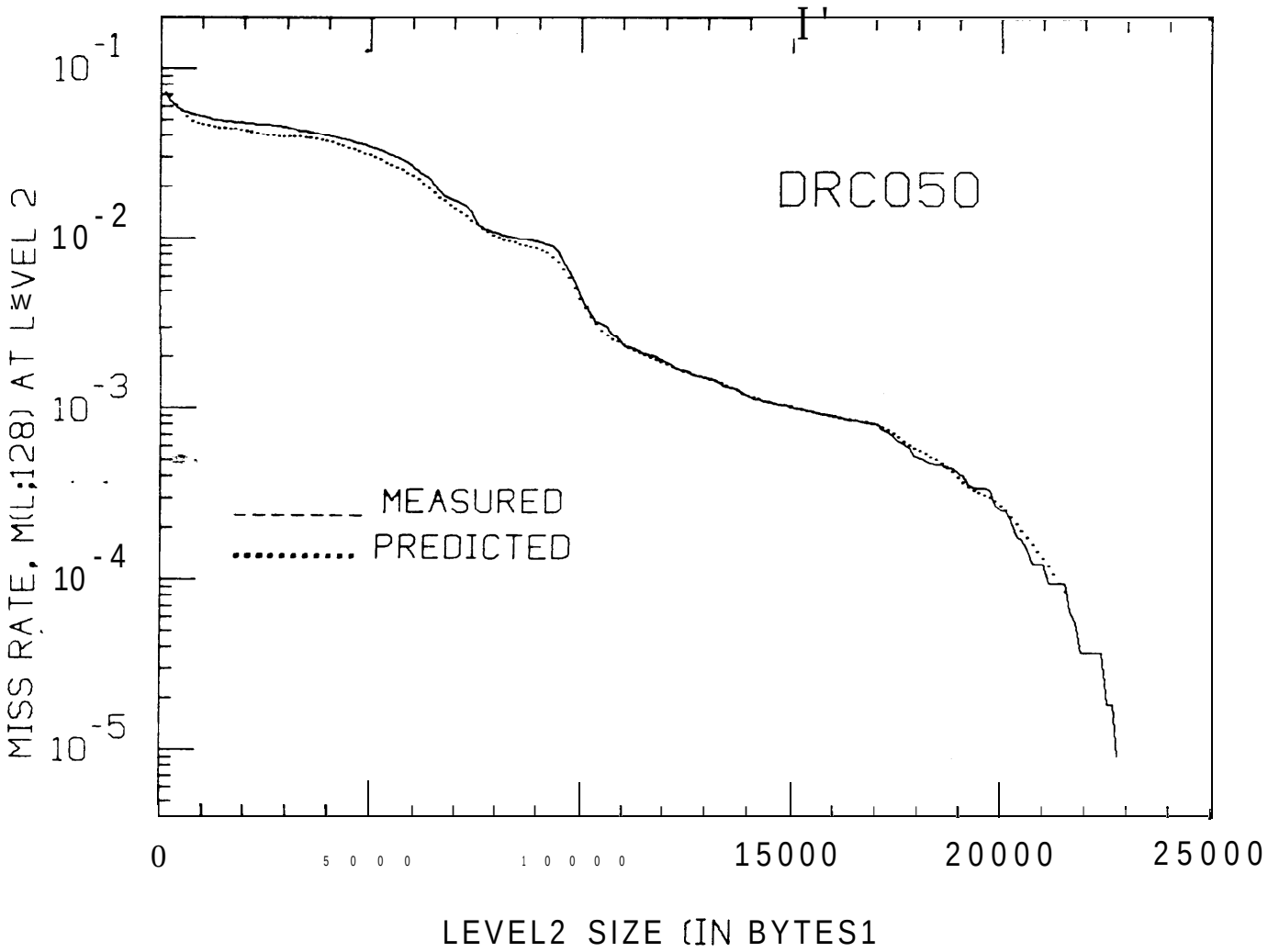

FIG 19

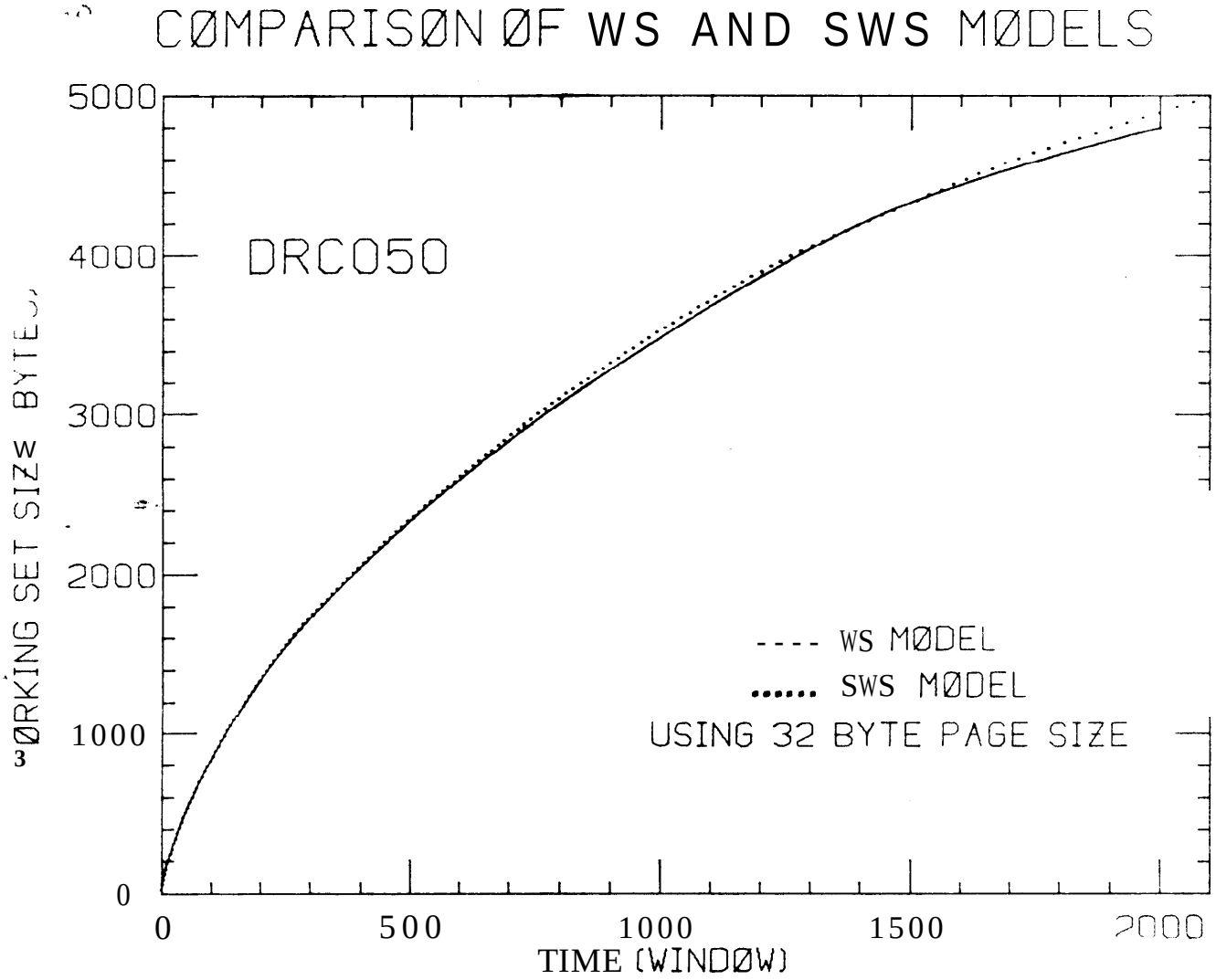
CØMPARISØN ØF WS AND SWS MØDELS
DRC050
5000
4000
3000
2000
1000
0
0
500
1000
1500
2000
TIME (WINDØW)
---- WS MØDEL
..... SWS MØDEL
USING 32 BYTE PAGE SIZE

FIG 20