Effect of Partially Preloaded Program on Paged Virtual Memory

YASUFUMI YOSHIZAWA* and MASATO KONDO**

A Partially Pre-Loaded Program (PPLP) approach is introduced in order to correct the incompleteness of the Page Replacement Algorithm (PRA) in paged virtual memory systems. In the PPLP approach, it is first necessary to determine the resident page set which may reduce the number of page faults. Based on an analysis of PPLP effects, four practical resident page selection algorithms are proposed. Address traces are used in PPLP experiments. The results are discussed with regard to the decrement of the page faults as well as the relation between program behavior and the selection algorithm.

1. Introduction

Many paged virtual memory computer systems have been developed to date. Most of them employ 'demand-paged memory management' in order to secure high level memory utilization. In such systems, the program's memory reference behavior has a great influence on computer performance as described in [1–2]. Accordingly, the operating system must always predict the program's memory reference behavior, and place pages which will be referred to in the near future in main memory to prevent page fault occurences. This function is known as 'Page Replacement Algorithm' (PRA) and is vital to paged virtual memory systems.

In general, an operating system should be developed assuming the hardware configuration and its application fields. Consequently, the system designers should employ a PRA which best suits their application. However, it is difficult for the PRA to adequately predict the program's memory reference behaviors. As a result, the PRA usually has some degree of prediction incompleteness in [3–4]. This PRA incompleteness for the frequently used programs such as compilers, some parts of supervisors, access methods and on-line programs causes great losses over long periods. For this reason, it is necessary to devise a way to correct PRA incompleteness.

The Partially Pre-Loaded Program (PPLP) approach presented in this paper, which makes a part of the program or data reside permanently in the main memory, is effective in correcting PRA incompleteness.

E. Gelenbe presented the effectiveness of the random partially preloaded page replacement algorithm in [5], and introduced the formula of page fault ratio for W. F. King's program model in [6] in which the referenced pages appear independently and obey the identical distribution. E. Gelenbe employed the random PRA in [3] to pageable area (An area of virtual storage whose addresses are not identical to real address), and he named this algorithm as Random Partially Preloaded (RPPL) PRA. In this situation, he found that the optimum resident page set contains m-1 pages whose probability of being referred is highest; where as, the main memory contains m pages.

However, we feel that the program's page reference on a real computer is not the independent, identically distributed model. J. R. Spirn and P. J. Denning were of the same opinion in [7], from the viewpoint of the program locality. Moreover, LRU (Least Recently Used) or Working Set PRA are widely adopted rather than Random or FIFO (First In First Out) PRA, especially in commercial computers.

In this paper, it is not necessary to assume a specified page reference model like the one used by W. F. King [6] for analysis of the PPLP approach. Moreover, the well-known LRU PRA is adopted. Thus, we aim to; (a) analyze the decrement of page fault number by PPLP, (b) propose the selection algorithms for the resident page set and (c) evaluate each selection algorithm. From the above considerations, we can find a resident page set which corrects the incompleteness of LRU PRA and reduces the number of page faults.

2. Analysis of Partially Pre-Loaded Program

2.1 Stack Processing

In the paged memory system, program behavior can be regarded as a page reference string, which is a sequence, $\omega = r_1 r_2, \ldots r_t, \ldots$, where $r_t \in N$, $t \geq 1$, and $i = r_t$ means that page i is referenced at the t-th reference; and a set of pages referred to by a program denote $N = \{1, 2, \ldots, n\}$.

Stack processing is an efficient technique for PRA evaluation, especially stack algorithms such as Least Recently Used (LRU), Least Frequently Used and optimal PRA reported in [8]. Therefore, stack processing shall be applied in PPLP analysis. The stack model can be described as follows: The top of the stack is current reference, second position is the next most recently

*Systems Development Laboratory, Hitachi Ltd.
**Former Systems Development Laboratory, Hitachi Ltd.

referred page, etc. When the page in stack position k is referenced, it is moved to the top; and all pages which were in position $1, 2, \ldots,$ and k-1 are pushed down one position. The stack distance of page x at the t-th reference is denoted by $d_t(x)=k$. If page x has not been referenced, distance $d_t(x)$ is set at infinity. Therefore, it is set in the first position and all pages are pushed one position.

If $d_t(x)>m$, a page fault occurs; and the page which was in position m is moved from the main memory to the secondary storage. That is the LRU PRA.

2.2 Optimal One Page for PPLP

In this section, consideration is given to what page will give the maximum effect if only one page is maintained in the main memory. To find the page, we shall define the number of page faults caused by page $x \in N$. The page fault function for page x at the t-th reference in ω is defined as;

$$\Delta_t(x, m)=\begin{cases}1 & \text{if } r_t=x \text{ and } d_t(x)>m,\\ 0 & \text{otherwise.}\end{cases} \tag{1}$$

Thus, the total number of page faults $D(x, m)$ caused by page x can be given by,

$$D(x, m)=\sum_{t=1}^{|\omega|}\Delta_t(x, m), \tag{2}$$

If page x is always retained in the memory, the page fault number seems to decrease $D(x, m)$. However, the pageable memory area loses one page; therefore, page $y \in N-\{x\}$ may cause page faults affected by the resident page x to occur.

Let us consider how many page faults will occur by $y \in N-\{x\}$ if page x is placed in the memory. Let page $y \in N-\{x\}$ be referred at the t-th reference in ω, and the position of page x in the stack be $d(x)$. If the distance $d(x)$ is less than or equal to m, page x does not affect page y. However, if $d(x)>m$ and $d_t(y)=m$, page y causes a page fault because one page of pageable area is lost. Therefore,

$$\Delta_t(x, y, m)=\begin{cases}1 & \text{if } r_t=y,\ d_t(y)=m \text{ and } d(x)>m,\\ 0 & \text{otherwise,}\end{cases} \tag{3}$$

expresses the page fault increment affected by resident page x. The total increment is obtained from eq. (3):

$$I(x, m)=\sum_{y\in N-\{x\}}\sum_{t=1}^{|\omega|}\Delta_t(x, y, m). \tag{4}$$

Due to eqs. (2) and (4), the total decrement of page faults according to resident page x is given by,

$$E(x, m)=D(x, m)-I(x, m). \tag{5}$$

Therefore, by using eq. (5), the optimal page x which resides in the memory and gives the maximum decrement of page faults, can be found.

2.3 Effect of the Resident Page Set

The previous arguments are generalized in this section. Let R denote the resident page set, where $R \subset N$ and $0 \le |R| \le m-1$, and consider the decrement number of page faults. If page set R is always maintained in the memory, the number of page faults caused by page $x \in R$ is given by,

$$D(R, m)=\sum_{x\in R} D(x, m). \tag{6}$$

On the other hand, the number of pages for the pageable area is $m-|R|$. This is expected to increase page faults caused by page $y \in \{N-R\}$ as described in the previous section. Therefore, stack status when page $y \in \{N-R\}$ is referred at the t-th reference in ω and its stack distance $d_t(y)=i$ are considered. Let $d(r)$ be the stack distance of page $r \in R$ in the stack. In this case, there are three stack conditions depending on $d(r)$ and $d_t(y)$. The three stack conditions are shown in Fig. 1.

(a) Case 1: $d(r)<d_t(y)$

Page y is not affected by page $r \in R$ whether page set R is placed in the memory or not.

(b) Case 2: $d(r)>d_t(y)$

All stack distances of page $r \in R$ are greater than $d_t(y)$;

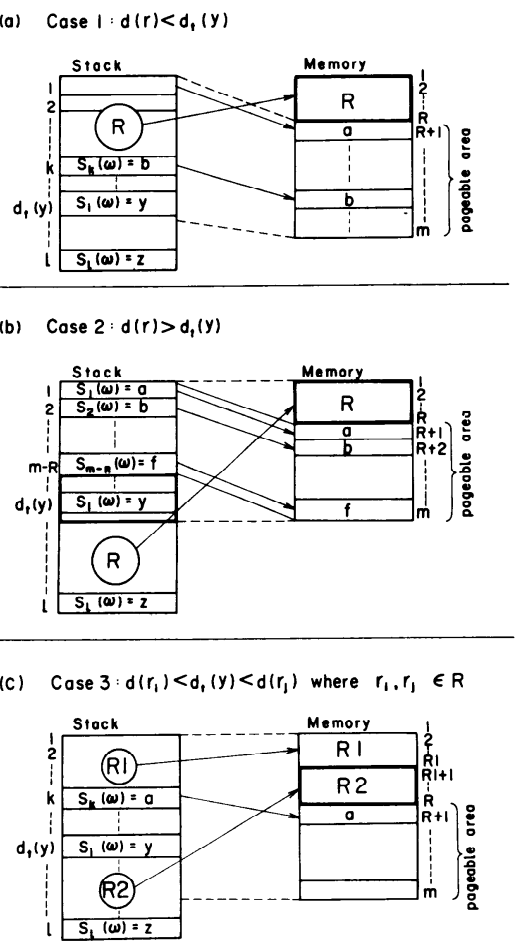

Fig. 1 The negative effect of resident page set R on page reference y.

thus, page set R has a great influence on the page reference y. If $d_t(y) \leq m-|R|$ or $d_t(y) > m$, page y is not affected. However, if $d_t(y)$ satisfies $m-|R| < d_t(y) \leq m$, page y is affected by R and causes a page fault. Memory utilization deteriorates if the page in lower positions of the stack are permitted to reside.

(c) Case 3: Stack distances $d(r)$ are greater or less than $d_t(y)$. That is, pages in R are split by means of $d_t(y)$ in the stack, as illustrated in Fig. 1. The condition $R = R1 \cup R2$ contains the above two cases; thus, case 1 and case 2 results can be used. It is evident that $R1$ is the same as case 1, because $d(r_1) < d_t(y)$ for $r_1 \in R1$. On the other hand, $R2$ affects reference y, if $d_t(y)$ satisfies

$$m-|R2| < d_t(y) \leq m \tag{7}$$

for $r_2 \in R2$, where $d(r_2) > d_t(y)$.

Case 3 is the most general; therefore, the page fault event caused by the resident page set R is defined as:

$$\Delta_t(R, y, m) = \begin{cases} 1 & \text{if } r_t = y,\ m-|R2| < d_t(y) \leq m, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Using eq. (8), the total number of page faults caused by R for $y \in \{N-R\}$ is given by,

$$I(R, y, m) = \sum_{t=1}^{|\omega|} \Delta_t(R, y, m). \tag{9}$$

Thus,

$$I(R, m) = \sum_{y \in \{N-R\}} I(R, y, m). \tag{10}$$

This is the sum of page faults increment caused by R for all $y \in \{N-R\}$.

Equations (6) and (10) make it possible to derive the effect of page faults decrement $E(R, m)$ caused by resident page set R. Thus

$$\begin{aligned} E(R, m) &= D(R, m) - I(R, m) \\ &= \sum_{x \in R} D(x, m) - \sum_{y \in \{N-R\}} I(R, y, m). \end{aligned} \tag{11}$$

Our aim is to find the page set R which makes eq. (11) maximum.

Let us consider eq. (11) terms. The first term $D(R, m)$ is a monotonously increasing function with respect to $|R|$. If we select $m-1$ pages whose $D(x, m)$ in eq. (2) is highest, the sum of $D(x, m)$ takes the maximum value. On the other hand, the second term is the number of page faults which is increased by resident page set R. Evidently, $I(R, m) = 0$ if $|R| = 0$. If the amount of resident pages is increased, stack status will become case 2 or case 3 in Fig. 1. And, stack distance $d_t(y)$ will come to have values which satisfy eq. (7). Thus, the second item in eq. (11) is a monotonously increasing function with respect to $|R|$.

From the above considerations, the number of page faults decrement $E(R, m)$ is the difference of monotonously increasing functions. It depends on the reference string and resident page set R.

3. Page Selection Algorithms for PPLP

3.1 Consideration of Optimal Resident Page Set R

The derived value $E(R, m)$ expresses the effect of resident page set R. This value is the difference between $D(R, m)$ and $I(R, m)$, which are closely connected to the page reference string and resident page set R. A page reference string is peculiar to the program; therefore, if the optimal resident page set R_0 is being sought, it is necessary to investigate $\sum_{r=1}^{m-1} \binom{n}{r}$ combination, where $|R| = r$. That is, the number of ways of selecting r of n distinct pages is $\binom{n}{r}$. For example, if $n = 10$ and $m = 5$. it is necessary to calculate $E(R, m)$ 386 times to search R_0. This involves an enormous amount of computation. For this reason, selection of R_0 is not practical. This presents the problem of proposing feasible selecting methods.

3.2 Stepwise Selection (SW) Method

Pages are successively selected as a resident set $R = \{x_1, x_2, \ldots, x_{m-1}\}$. The k-th page gives the maximum page faults decrement when the memory page is $m-k+1$, and a page string does not contain $\{x_1, x_2, \ldots, x_{k-1}\}$. That is, x_k satisfies $E(x_k, m-k+1) = \max E(x, m-k+1)$ in eq. (5). This approach is called the 'SW method'.

The cumulative decrement of page faults when k pages are placed in the memory is defined as:

$$E(k) = \sum_{i=1}^{k} E(x_i, m-i+1). \tag{12}$$

Let r_0 be the number of pages which makes eq. (12) maximum. Thus, $R_0(SW) = \{x_1, x_2, \ldots, x_{r_0}\}$ is the optimal resident page set in the SW method. Notice that PPLP is not useful if $E(r_0)$ is negative.

3.3 Working Set (WS) Method

In LRU PRA, the order of page reference is reflected in page replacement instead of time history. Therefore, page faults will occur frequently in programs which contain a large frequently used loop L, and which also contain a subroutine that refers many pages in a short time after execution of the loop L and return to the loop L.

Consequently, it is considered important in LRU PRA to supply information about the time history of a page reference. In order to measure the time history of the reference for each page over ω, the existent ratio in the working set $W(t, T)$ is sought. So, the existent ratio in $W(t, T)$ for a loop L must be high. Therefore, it is felt that a resident loop L will decrease page fault occurences.

Let $\omega = r_1 r_2 \ldots, r_k$ be a page reference string of length $k < \infty$, where the index of the references means instruction steps. Define the case where page x_t is in the working set $W(t, T)$ as:

$$\Delta_t(x_i, T)=\begin{cases}1 & \text{if } x_i=r_t \text{ and } x_i\in W(t, T),\\ 0 & \text{otherwise.}\end{cases}$$

The working set $W(t, T)$ at time t, for $T\geq 1$, is defined as the set of distinct pages referenced in the interval $[t-T+1, t]$ for $T\leq t$, or $[1, t]$ for $T>t$ as defined in [9] by P. J. Denning. This is called T window size. Thus, the existent ratio of x_i in $W(t, T)$ over ω is given by,

$$e_i(T)=\frac{1}{|\omega|}\sum_{t=1}^{|\omega|}\Delta t(x_i, T). \tag{13}$$

A resident page set R_j(WS) may be found by selecting j pages according to decreasing $e_i(T)$ in eq. (13). This page selection method is called the 'WS method'. The optimal resident page set R_0(WS) makes $E(R_j$(WS), $m)$ maximum.

To retain the page set R_j(WS) constantly in memory means that there are many instances of case 1 rather than case 2 or case 3 in Fig. 1. Consequently, it is felt that page faults of nonresident page $y\in\{N-R\}$ affected by R_j(WS) are not numerous. Further, if the stack status temporarily takes case 2 or case 3 in Fig. 1, useless page in/out from/to a secondary storage can be prevented. This is expected to reduce page faults because pages in R_j(WS) will be referred to in the near future with high probability. In the sequel, the WS method is very useful for a program whose time variant working set size $|W(t, T)|$ is large.

3.4 Page Reference Frequency (FR) Method

Let f_i be the frequency of references for a page i over a page reference string. Resident page set R_j(FR) contains j pages according to decreasing f_i. By calculating $E(R_j$(FR), $m)$ where $1\leq j\leq m-1$, it is possible to find the optimal resident page set R_0(FR) that makes $E(R_j$(FR), $m)$ maximum.

It is presumed that the pages whose reference frequency are high have a high existent ratio in $W(t, T)$ and reside in a higher part of the stack.

3.5 Sum of Stack Distance (SD) Method

If the stack distance $d_t(x)$ of page x at the t-th reference is larger than memory size m, a page fault occurs. Thus, the sum of the stack distances for each page seems to have the potential for page fault occurence. Resident page set R_j(SD) contains j pages according to the decreasing sum of stack distances.

This method aims to maximize the first term in eq. (11), but does not consider the second term.

4. Experimental Results and Evaluations

4.1 Program Behavior Analysis

Data traced on a real, running system have been used in an experimental study of PPLP approach, with the following objectives: (a) to confirm the effect of page fault decrement in PPLP approach; (b) to evaluate four algorithms which are proposed in the previous section; (c) to know the relationship between program behavior and resident page selection algorithm.

Table 1 Summary of the programs.

Name	A	B
Size (number of pages*)	27	30
Number of instructions	1,350,487	1,884,150
Length of page string	58,783	159,160

*page=4,096 bytes

The trace data contain instruction and operand addresses faithfully. Therefore, the page reference string can be obtained from the trace data. It is assumed that a page size is 4,096 bytes. Two programs have been used in the experiments. The main characteristics of two programs are summarized in Table 1.

Page reference frequency f_i, existent ratio $e_i(T)$ in $W(t, T)$, and the sum of the stack distance were examined for each page from the page string. From these results, resident page sets were made, R_j(WS), R_j(SW), R_j(FR), R_j(SD) where $1\leq j\leq m-1$; and the decrement number of page faults were calculated by using the LRU simulator. Through the 'SW method', a program was developed to automatically generate resident page set R_j(SW) and calculate $E(R_j$(SW), $m)$ in eq. (11).

4.2 Comparison of Each Selection Algorithm

(i) Measurement of PPLP

By using eq. (11), the number of page fault occurences is determined. Then,

$$\begin{aligned}F(R, m)&=\sum_{x\in R}D(x, m)-E(R, m)\\&=\sum_{x\in\{N-R\}}\{D(x, m)+I(R, x, m)\}.\end{aligned} \tag{14}$$

Let $F(0, m)$ be the number of page faults where $r=0$. Define the effect of PPLP as:

$$\rho(R, m)=\frac{F(0, m)}{F(R, m)}. \tag{15}$$

The effect $\rho(R, m)>1$ means that PPLP is useful. Let $g(R, m)$ be the decreasing ratio of page fault. Thus, using eq. (15),

$$g(R, m)=\left\{1-\frac{1}{\rho(R, m)}\right\}*100(\%). \tag{16}$$

(ii) Variance of page faults with resident page set

The number of page faults $F(R, m)$ is peculiar to a resident page set R. The variance of page faults is shown in Fig. 2 as a function of the page resident set R_j(SW) for the program B. Consider the function $D(x, m)$ and $I(R, x, m)$ in eq. (14). As described in section 2.3, the function $I(R, x, m)$, which is the number of page faults affected by R, is a monotonously increasing function with respect to $|R|$. Thus, $F(R, m)$ is the sum of the monotonously increasing and decreasing functions.

Notice that Fig. 2 shows that the number of page faults decrease to an extent, namely $4\leq m\leq 14$. That is,

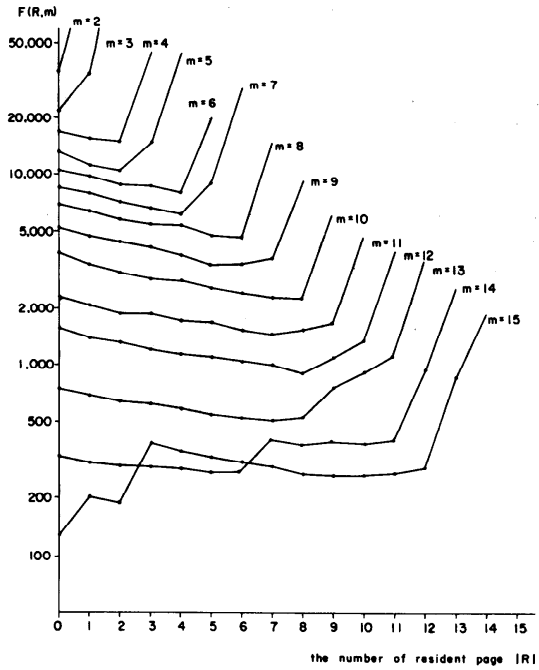

Fig. 2 Page fault function $F(R, m)$ for program B with algorithm SW.

there is an effective range of memory size for PPLP. With small memories, if resident pages are selected, the pageable area becomes extremely small. Consequently, resident page set R affects pages $y \in \{N-R\}$ and page y causing page faults to occur. Therefore, $E(R, m)$ in eq. (11) becomes negative. On the other hand, the sum of $I(R, x, m)$ for $y \in \{N-R\}$ becomes greater than the sum of $D(x, m)$ for $x \in R$ in large memories; thus, $E(R, m)$ in eq. (11) is also negative.

(iii) Evaluation of resident page selection algorithms

Four algorithms were tested for program A and B. The results for A and B are presented in Fig. 3 and Fig. 4, respectively. These figures show that the effect of PPLP is a function of memory size m. In an approach to the working set (WS method), a window size of 1,000 instruction steps was taken for program A and 20,000 steps for program B. These window sizes gave the maximum effect in this approach for the program.

The effect of PPLP, $\rho(R, m)$, is a function of memory size m and resident page set R. Fig. 3 shows that the WS method is the best, and the improvement is at most $g(R, m)=97\%(\rho(R, m)=30.12)$, where $m=13$ and $|R_0(\mathrm{WS})|=6$. The SW method is the second best; and the improvement is, at most, $g(R, m)=65\%(\rho(R, m)=2.83)$, where $m=13$ and $|R_0(\mathrm{SW})|=9$. The FR and SD methods indicate almost the same results, and the effect being, at most, $g(R, m)=40\%(\rho(R, m)=1.66)$.

Fig. 4 shows that the SW method is always better than other algorithms for program B. The maximum

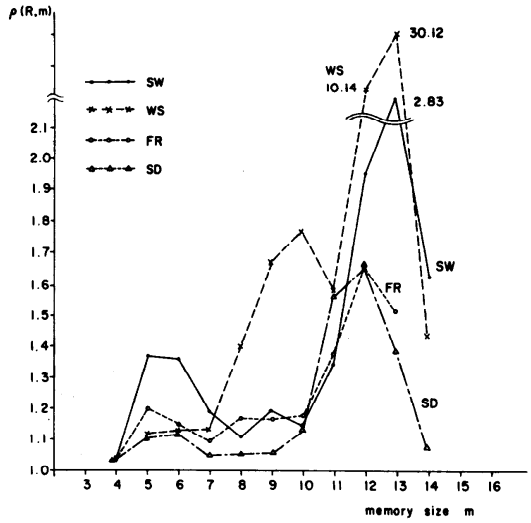

Fig. 3 Effect of partially preloaded program for program A.

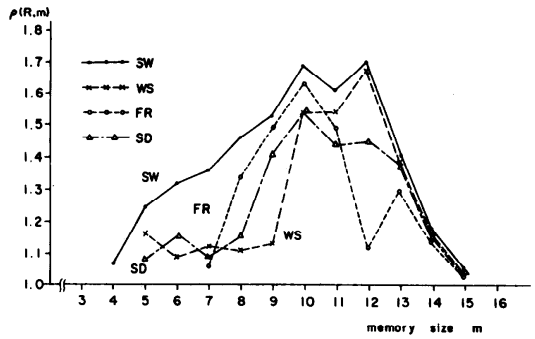

Fig. 4 Effect of partially preloaded program for program B.

improvement of the SW method is $g(R, m)=41\%(\rho(R, m)=1.7)$, where $m=12$ and $|R_0(\mathrm{SW})|=9$. The WS method is the second best, and slightly better than the FR method. The SD method is the worst; however, the improvement is $g(R, m)=35\%(\rho(R, m)=1.54)$, where $m=10$ and $|R_0(\mathrm{SD})|=8$.

In section 3.3, it was stated that the WS method has a very powerful effect on the programs whose time variant working set size $|W(t, T)|$ is large. Consequently, the working set size for programs A and B were investigated. The results are shown in Fig. 5. Evidently, the working set size of program A is very variant; however, program B's working set size is almost stable with respect to window size T.

5. Conclusion

There are many operating systems which basically adopt LRU or Working Set strategy PRA. In this paper, methods of reducing the page fault occurences with PPLP were analyzed. The number of page faults reduced by the resident page set R is given by $E(R, m)$ in

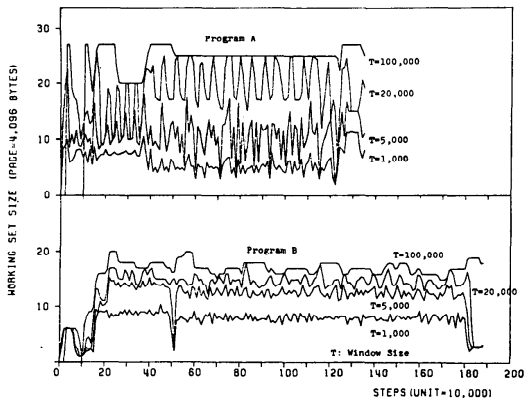

Fig. 5 Time variance of the working set size $|w(t, T)|$ for program *A* and *B*.

eq. (11), where the number of memory pages is m.

It is necessary to calculate $\sum_{r=1}^{m-1} \binom{n}{r}$ times in order to get the optimum resident page set R_0 which gives maximum $E(R, m)$, where the number of reference page is n and $r=|R|$. However, this approach is not practical. Therefore, practical and feasible resident page selection algorithms are proposed; namely, SW (stepwise), WS (higher existent ratio in the working set), FR (page reference frequency) and SD (sum of stack distance). In evaluating those algorithms, trace data were used which were on real, running systems (and converted to page reference strings) to investigate the program behavior for algorithms. Thus, an LRU simulator was developed and each algorithm was compared with regard to the degree of page faults decrement.

From the experimental study, it was confirmed that the PPLP approach is a very useful and practical strategy in decreasing page fault in paged virtual memory systems. The WS method was found to be especially powerful with programs whose time variant working set size is large. Moreover, the SW method was found to be suitable for programs whose working set size is steady with respect to window size. During the course of several experiments, the WS and SW methods showed better improvement than the FR and SD methods. The FR and SD methods show almost the same properties.

Acknowledgment

The author is grateful to Taizo Nauchi of the Software Works, Hitachi, Ltd. for many useful suggestions relating to PPLP approach.

References

1. DENNING, P. J. Thrashing: Its causes and prevention. *Proc. of FJCC* **33**, (1968), 915–922.
2. HATFIELD, D. J. AND GERALD, J. Program Restructuring for Virtual Memory. *IBM Systems Journal* **10**, 3, (1968), 168–192.
3. BELADY, L. A. A Study of Page Replacement Algorithms for a Virtual Storage Computer. *IBM Systems Journal* **5**, 2, (1966), 78–101.
4. PRIEVE, B. G. AND FABRY, R. S. VMIN-An Optimal Variable-Space Page Replacement Algorithm. *CACM*, **19**, 5 (May 1976), 295–297.
5. GELENBE, E. A Unified Approach to the Evaluation of Replacement Algorithm. *IEEE Trans. on Computers*, c-22, 6 (June 1973).
6. KING III, W. F. Analysis of Paging Algorithms. *Proc. of IFIP 71 Congr*, 1971.
7. SPIRN, J. R. AND DENNING, P. J. Experiments with Program Locality, *Proc. of FJCC* **41**, (1972), 611–621.
8. MATTSON, R. L., GECSEI, J., SLUTZ, D. R. AND TRAIGER, I. L. Evaluation Techniques for Storage Hierarchies. *IBM Systems Journal*, **9**, 2, (1970), 78–117.
9. DENNING, P. J. The Working Set Model for Program Behavior. *CACM*, **11**, 5 (May 1968), 323–333.