# Dynamic Working-Set Memory Allocation for Concurrent Processes

*By Emile Haddad*

TR 90-3

# DYNAMIC WORKING-SET MEMORY ALLOCATION FOR CONCURRENT PROCESSES

EMILE HADDAD

TR 90-3

# DYNAMIC WORKING-SET MEMORY ALLOCATION FOR CONCURRENT PROCESSES

Emile K. Haddad, Ph.D.
Department of Computer Science
Virginia Polytechnic Institute and State University
2990 Telestar Court, Falls Church, Virginia 22042

Abstract -- The problem of allocating a given total amount of main memory page-frames M(t), available at time t, among a given set of concurrently active processes $\{P_i\}$ is examined. Memory allocation is managed under the working set policy, and it is assumed that M(t) is larger than the total sum of all working-set sizes but less than the sum of distinct pages in $\{P_i\}$. It is required that each process receive at least an allocation equal to its working-set size and that the excess memory be additionally distributed among the concurrent processes to enhance their performance. The upper and lower bounds on the individual memory allocations for each process are derived. A fair policy for apportioning the excess available memory among the concurrent processes is presented. A procedure for rounding off fragmented page apportionments is described. An application example, illustrating dynamically changing conditions in the concurrently active processes, is provided.

## Introduction and Problem Formulation

Consider a paged memory system which supports a number n(t) of concurrently active processes at any given time t. Denote the group of processes by

$$\{P_i\} \quad , \quad i \in I(t) \subset \{1,2,...\} \quad , \quad |I(t)| = n(t) \tag{1}$$

where $|I|$ denotes the cardinality of the set I. The value n(t) fluctuates as new active processes join the group $\{P_i\}$ and existing active processes are suspended or terminated. The total amount of main memory available for the group is M(t) page-frames, which has to be apportioned among the individual processes, with $P_i$ allocated $m_i(t)$ memory frames, such that

$$\Sigma_{i \in I(t)} m_i(t) = M(t) \, . \tag{2}$$

For each $P_i$, $m_i(t)$ resident pages are kept in main memory at time t. The memory management scheme for the group $\{P_i\}$ is based on the working-set policy, which requires that the set of resident pages for $P_i$ should, at all times, include all pages of the working-set $W_i(t,d_i)$ defined as the set of distinct pages referrenced by $P_i$ during the $d_i$ most recent references [1]. The parameter $d_i$ is a positive integer referred to as the working-set window. Let $w_i$ denote the cardinality, i.e., number of pages, of the set $W_i$

$$w_i(t,d_i) = |W_i(t,d_i)| \, . \tag{3}$$

Note that at all times during the execution of process $P_i$ one has

$$1 \leq w_i(t,d_i) \leq d_i(t) \qquad , \text{all t} \, . \tag{4}$$

The parameter $d_i$, which ordinarily is a specified constant, is represented in (4) as a function of time in order to cover the possibility of a dynamic memory management scheme where $d_i$ is allowed to change during the execution of $P_i$. Let $p_i$ be the total number of distinct pages referenced by process $P_i$. Evidently one has

$$1 \leq w_i(t,d_i) \leq d_i(t) \leq p_i \, . \tag{5}$$

The working-set memory management policy [2], [3] prescribes that

$$m_i(t) \geq w_i(t,d_i) \qquad , \text{all t} \, . \tag{6}$$

This means that $w_i$ is the minimum size that at all times should be maintained for the resident set. Larger values of $m_i$ will result in improved performance as shown in the typical representation of Figure 1. Evidently, there is no benefit or improvement in performance to be gained from assigning to $P_i$ memory frames in excess of $p_i$, and we should therefore require that $m_i \leq p_i$

$$w_i(t,d_i) \leq m_i(t) \leq p_i \qquad \text{, all } t \text{ .} \tag{7}$$

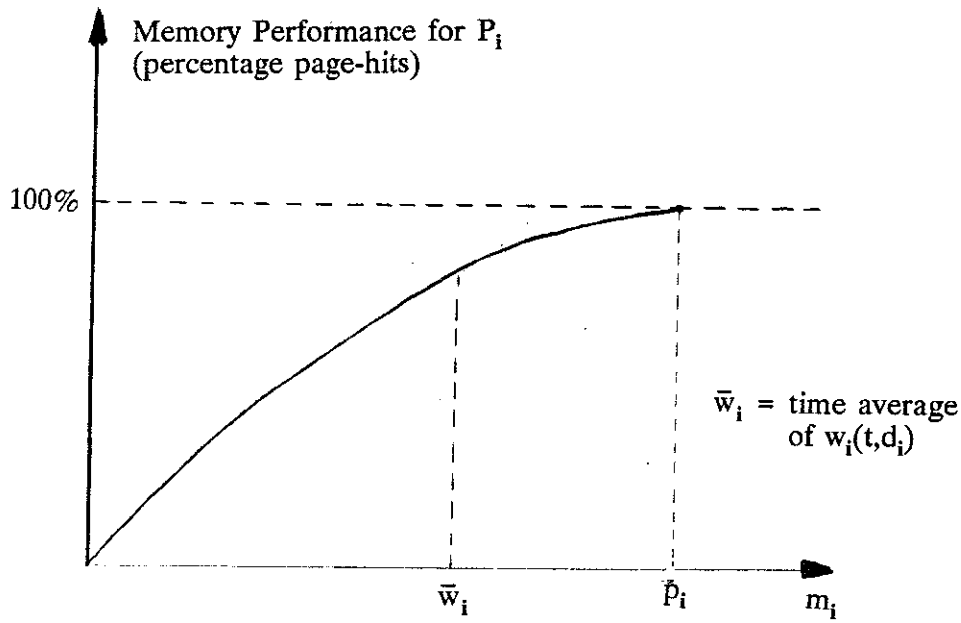


Figure 1. Memory Performance versus Resident-Set Size

The requirement in (2) is combined with (7) to give

$$\Sigma_{i\in I(t)} m_i(t) = M(t) \qquad \text{, all } t \tag{8}$$

$$\Sigma_{i\in I(t)} w_i(t,d_i) \leq M(t) \leq \Sigma_{i\in I(t)} p_i \qquad \text{, all } t \text{ .} \tag{9}$$

This paper poses the following question: How should the values of $m_i$ be chosen subject to the requirements in (7), (8), and (9), where $w_i$, $p_i$, and M are given parameters.

## Memory Allocation

In the interest of simplifying the notation for the remaining analysis, we shall consider the dependence on time of the various terms in (7), (8), and (9) to be understood and drop their explicit dependence on the variable t:

$$w_i \le m_i \le p_i \qquad \text{, all } i \in I \tag{10}$$

$$\Sigma_{i\in I} m_i = M \tag{11}$$

$$\Sigma_{i\in I} w_i \le M \le \Sigma_{i\in I} p_i \,. \tag{12}$$

Recall the significance of the condition in (12) in relationship to the load control of the group of active processes $\{P_i\}$. The left side inequality $\Sigma w_i \le M$ implies that M should be sufficiently large to provide each $P_i$ with at least $w_i$ memory frames, the size of its working set. If at any time this becomes untrue, i.e., if $\Sigma w_i > M$, the load controller would reduce the number $|I|$ of active processes by suspending some processes so that the summation of $w_i$ for the remaining active processes is less than M. On the other hand, if at any time one has $M \ge \Sigma p_i$, the load controller would attempt to activate additional processes, if any are available, so that (12) is satisfied. If there are no additional processes queued for activation, the load controller would assign $m_i = p_i$ for all i, with a resulting optimum performance for each process, as indicated in Figure 1. These situations aside, the problem addressed by this paper is represented by the situation where the available M is larger than the sum of the working set sizes of the active processes, and there are no additional processes to be activated, and the *excess* memory frames $E = M - \Sigma w_i$ is to be distributed among the currently active processes for *improved performance*. Let $e_i$ denote the excess over $w_i$ of memory frames allocated to $P_i$,

$$e_i \equiv m_i - w_i \quad , \quad \Sigma_{i\in I} e_i = E = M - \Sigma_{i\in I} w_i \,. \tag{13}$$

This task of apportioning E among the group of concurrently active processes is not always as straightforward as it might seem on a first look. We shall shortly show that the conditions in (10)-(12) imply that each $m_i$ is restricted to an interval of values with a maximum of $\hat{m}_i \le p_i$ and

a minimum $\tilde{m}_i \geq w_i$ whose values will be determined in terms of the given parameters. In other words, the conditions in (10)-(12) constrain each $m_i$ to an interval which is a subset of the interval indicated in (10):

$$m_i \in [\tilde{m}_i, \hat{m}_i] \subseteq [w_i, p_i] \ . \tag{14}$$

The condition in (14) implies that, irrespective of the criterion adopted for apportioning the excess memory frames E among the concurrently active processes $\{P_i\}$, the share $e_i$ for $P_i$ should not be less than $\tilde{m}_i - w_i$ and should not be greater than $\hat{m}_i - w_i$

$$\tilde{m}_i - w_i \leq e_i \leq \hat{m}_i - w_i \ . \tag{15}$$

Thus a knowledge of the values of $\tilde{m}_i$ and $\hat{m}_i$ is necessary before one can apportion the excess memory E into the various individual allocations among the concurrent processes $\{P_i\}$. The memory allocation policy might, for example, prescribe distributing E equally among the active processes, or proportionately to their size, or according to some other criterion. In all cases, however, this should be done without violating the requirement in (15). In the following section, we shall derive the general expressions for the memory allocation bounds $\tilde{m}_i$ and $\hat{m}_i$ in terms of all the other parameters in (10)-(12), namely $w_i$, $p_i$, M, and I. We shall then proceed to prescribe a *fair* policy for dividing E among the active group of concurrent processes.

## Memory Allocation Bounds

We now formalize the definitions of $\hat{m}_i$ and $\tilde{m}_i$: the upper and lower bounds on $m_i$. Let $m = \{m_i\}$ be any set of numbers $m_i$ that satisfies (10)-(11):

$$m = \{m_i\} \ , \ i \in I \ , \ w_i \leq m_i \leq p_i \ , \ \Sigma m_i = M \ . \tag{16}$$

Let C be the set of all m that satisfy (16)

$$C = \left\{ m = \{m_i\} \ , \ m_i \text{ integer} \ , \ i \in I \ : \ w_i \leq m_i \leq p_i \ , \ \Sigma m_i = M \right\} \ . \tag{17}$$

Our interest is to determine, for any given i, the smallest and largest values of $m_i$ as m takes all possible values in the set C

$$\tilde{m}_i = \min_{m \in C} m_i \quad , \quad \hat{m}_i = \max_{m \in C} m_i \quad , \quad i \in I \, . \tag{18}$$

The statement in (18) is equivalent to saying that, for any given $i \in I$, there exists a specific set $m = \mu \in C$ whose $i^{th}$ element is equal to $\tilde{m}_i$ and a specific set $m = \gamma \in C$ whose $i^{th}$ element is equal to $\hat{m}_i$, and that the $i^{th}$ element of all other $m \in C$ lie in the interval $[\tilde{m}_i, \hat{m}_i]$:

$$\mu \in C \quad , \quad \mu_i = \tilde{m}_i \quad , \quad \gamma \in C \quad , \quad \gamma_i = \hat{m}_i \tag{19}$$

$$\tilde{m}_i \leq m_i \leq \hat{m}_i \qquad \text{for all } m \in C \, . \tag{20}$$

From (19) and (17) it follows that

$$w_i \leq \tilde{m}_i \leq p_i \quad , \quad w_i \leq \hat{m}_i \leq p_i \, . \tag{21}$$

Combining (21) with (20), we obtain

$$w_i \leq \tilde{m}_i \leq m_i \leq \hat{m}_i \leq p_i \qquad \text{all } m \in C \, . \tag{22}$$

Our objective is to determine $\tilde{m}_i$ and $\hat{m}_i$ in terms of the given parameters $w_i$, $p_i$, and M; and to answer the following questions:

1. Under what conditions is $\tilde{m}_i = w_i$ and $\hat{m}_i = p_i$?
2. Under what conditions is $\tilde{m}_i > w_i$ and $\hat{m}_i < p_i$, and if so, what are the values of $\tilde{m}_i$ and $\hat{m}_i$ in terms of the given parameters.

The following Theorem provides the complete answer to these questions.

Theorem

For any $i \in I$, the values of $\hat{m}_i$ and $\tilde{m}_i$ are determined as follows:

(1) If $p_i + \Sigma_{j \neq i} w_j \leq M$, then $\hat{m}_i = p_i$ ; otherwise (23)

if $p_i + \Sigma_{j \neq i} w_j > M$, then $\hat{m}_i = M - \Sigma_{j \neq i} w_j < p_i$ . (24)

(2) If $w_i + \Sigma_{j \neq i} p_j \geq M$, then $\tilde{m}_i = w_i$ ; otherwise (25)

if $w_i + \Sigma_{j \neq i} p_j < M$, then $\tilde{m}_i = M - \Sigma_{j \neq i} p_j > w_i$ . (26)

The proof of the Theorem is given in the Appendix.

## Fair Memory Allocation

The foregoing analysis has shown that, subject to requirements of the memory allocation problem as stated in (10)-(12), each individual allocation $m_i$ must be chosen such that its value is confined between an upper bound $\hat{m}_i$ and a lower bound $\tilde{m}_i$ whose values are determined by the Theorem. Thus $m_i$ may vary over the interval $[\tilde{m}_i,\hat{m}_i]$, and its value may be expressed as follows:

$$m_i = \tilde{m}_i + r_i(\hat{m}_i - \tilde{m}_i) \quad , \quad 0 \leq r_i \leq 1 \, . \tag{27}$$

Paraphrasing the expression in (27), one may say that process $P_i$ is allocated its minimum admissible allocation $\tilde{m}_i$ plus a certain fraction r of its total range of variation $\hat{m}_i - \tilde{m}_i$. Since memory performance for process $P_i$ improves as $r_i$ is increased, we shall prescribe a *fair* allocation policy by requiring the fraction $r_i$ to have the *same* value r for *all* processes $P_i$. Setting $r_i = r$ in (27), we obtain

$$m_i = \tilde{m}_i + r(\hat{m}_i - \tilde{m}_i) \quad , \quad 0 \leq r \leq 1 \quad \text{all } i \in I \, . \tag{28}$$

The exact numerical value of r may now be uniquely determined by invoking the requirement $\Sigma m_i = M$ as stated in (11):

$$M = \Sigma_{i\in I}\tilde{m}_i + r\Sigma_{i\in I}(\hat{m}_i - \tilde{m}_i) \tag{29}$$

$$r = (M - \Sigma_{i\in I}\tilde{m}_i)/\Sigma_{i\in I}(\hat{m}_i - \tilde{m}_i) \, . \tag{30}$$

To verify that the value of r thus found does in fact satisfy the requirement $0 \leq r \leq 1$ as stated in (28), note that (22) implies the following:

$$\Sigma_{i\in I}(\hat{m}_i - \tilde{m}_i) \geq \Sigma_{i\in I}(m_i - \tilde{m}_i) = M - \Sigma\tilde{m}_i \geq 0 \, . \tag{31}$$

Combining (31) with (30), one has $0 \leq r \leq 1$. To recapitulate, the procedure for determining the fair allocation is:

1. Determine $\tilde{m}_i$ and $\hat{m}_i$ for each $i \in I$ using the expressions (23)-(26) in the Theorem.
2. Determine the value of r from (30).
3. Determine each $m_i$ from (28).

It should be noted that the value of r as obtained from (30) may not be an integer, and therefore the term $r_i(\hat{m}_i - \tilde{m}_i)$ in (28) may not be an integer, in which case $m_i$ would have a noninteger value. Although this is theoretically acceptable, practical implementation considerations would ordinarily dictate a memory management policy that prescribes nonfractional allocation of page frames [4]. This would require that the values of $m_i$ obtained by the above allocation policy be rounded to integer values. This should be carried out in such a way as to preserve the requirement $\Sigma m_i = M$ for the rounded values. The following is an adequate procedure for that purpose:

1. Let $\lfloor m_i \rfloor$ denote the largest integer not greater than $m_i$.
2. Let $\bar{m}_i$ denote the rounded value of $m_i$, viz $\bar{m}_i = \lfloor m_i \rfloor$ if $m_i$ is rounded down and $\bar{m}_i = \lfloor m_i \rfloor + 1$ if $m_i$ is rounded up.
3. Let

$$\eta = \Sigma_{i \in I}(m_i - \lfloor m_i \rfloor) = M - \Sigma_{i \in I} \lfloor m_i \rfloor \tag{32}$$

be the *integer* representing total "deficiency" resulting from rounding down all $m_i$. Note that $\eta$ is an integer because M and $\lfloor m_i \rfloor$ are integers, and that $\eta < n = |I|$ because $m_i - \lfloor m_i \rfloor < 1$. The integer value $\eta$ has to be distributed among $\eta$ values of $\lfloor m_i \rfloor$, each being rounded up by receiving an additional 1. These $\eta$ values of $m_i$ to be rounded up are chosen as follows.

4. Arrange the values of $m_i$ in a one-dimensional array ordered by the nonincreasing values of $m_i - \lfloor m_i \rfloor$:

$$m_{i_1}, m_{i_2}, \ldots, m_{i_j}, \ldots, m_{i_n} \qquad i_j \in I \tag{33}$$

$$m_{i_{(j+1)}} - \lfloor m_{i_{(j+1)}} \rfloor \leq m_{i_j} - \lfloor m_{i_j} \rfloor \qquad \text{for all } 1 \leq j \leq n \tag{34}$$

where $i_1, i_2, \ldots, i_n$ are the elements of I.

5. The first $\eta$ elements of the array in (33) are rounded up, and the remaining elements are rounded down

$$\bar{m}_{i_j} = \lfloor m_{i_j} \rfloor + 1 \quad \text{for all } 1 \le j \le \eta \tag{35}$$

$$\bar{m}_{i_j} = \lfloor m_{i_j} \rfloor \quad \text{for all } \eta < j \le n . \tag{36}$$

Note that from (35), (36), and (32), we obtain

$$\Sigma_{j=1}^{n} \bar{m}_{i_j} = \Sigma_{j=1}^{\eta} (\lfloor m_{i_j} \rfloor + 1) + \Sigma_{j=\eta+1}^{n} \lfloor m_{i_j} \rfloor = \eta + \Sigma_{j=1}^{n} \lfloor m_{i_j} \rfloor = M$$

which verifies that, as required, the rounded allocations add up to M. Moreover, the rounding procedure described above is *fair* because it selects for rounding up those values of $m_i$ that are closest to the rounded-up value $\lfloor m_i \rfloor + 1$, i.e., the furthest from the rounded-down value $\lfloor m_i \rfloor$, as indicated by the process of constructing the array in (33) and (34). To illustrate the above procedure with a specific numerical example, suppose the values of $m_i$ obtained by the allocation of M=30 among seven processes are

$$4.4 + 2.1 + 3.2 + 5.3 + 1.9 + 3.6 + 9.5 = 30$$

for which the value of $\eta$ is

$$\eta = 30 - \Sigma \lfloor m_i \rfloor = 30 - 27 = 3 .$$

Arranging the $m_i$'s in an array as described by (33) and (34), we have

$$1.9 , 3.6 , 9.5 , 4.4 , 5.3 , 3.2 , 2.1 .$$

The rounded values $\bar{m}_i$ are obtained by rounding the first three elements up and the remaining elements down:

$$\{\bar{m}_i\} = 2 , 4 , 10 , 4 , 5 , 3 , 2 .$$

Note that $\Sigma \bar{m}_i = M = 30$, i.e., the rounding preserved the value of the sum.

## Example

We now illustrate the application of the foregoing results in the dynamic memory allocation for a group of concurrent processes in a system that employs the working-set memory management scheme. Table 1 presents the given parameters as well as the computed results of the proposed memory allocation policy for four instants of time $t_1$, $t_2$, $t_3$, $t_4$.

Table 1. Examples

| $t_k$ | $t_1$ | | | | $t_2$ | | | | $t_3$ | | | | $t_4$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M(t_k)$ | 24 | | | | 24 | | | | 26 | | | | 26 | | |
| $\{P_i\}$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_1$ | $P_2$ | $P_5$ | $P_6$ | $P_2$ | $P_5$ | $P_6$ |
| $p_i$ | 4 | 6 | 8 | 10 | 4 | 6 | 8 | 10 | 4 | 6 | 10 | 20 | 6 | 10 | 20 |
| $w_i(t_k)$ | 2 | 3 | 2 | 4 | 3 | 5 | 7 | 2 | 2 | 3 | 5 | 5 | 4 | 8 | 2 |
| $\tilde{m}_i$ | 2 | 3 | 4 | 6 | 3 | 5 | 7 | 6 | 2 | 3 | 5 | 6 | 4 | 8 | 10 |
| $\hat{m}_i$ | 4 | 6 | 8 | 10 | 4 | 6 | 8 | 9 | 4 | 6 | 10 | 16 | 6 | 10 | 14 |
| r | $\frac{9}{13}$ | | | | $\frac{1}{2}$ | | | | $\frac{1}{2}$ | | | | $\frac{1}{2}$ | | |
| $m_i$ | $3\frac{5}{13}$ | $5\frac{1}{13}$ | $6\frac{10}{13}$ | $8\frac{10}{13}$ | $3\frac{1}{2}$ | $5\frac{1}{2}$ | $7\frac{1}{2}$ | $7\frac{1}{2}$ | 3 | $4\frac{1}{2}$ | $7\frac{1}{2}$ | 11 | 5 | 9 | 12 |
| $\eta$ | 2 | | | | 2 | | | | 1 | | | | 0 | | |
| $\bar{m}_i$ | 3 | 5 | 7 | 9 | 4 | 6 | 7 | 7 | 3 | 5 | 7 | 11 | 5 | 9 | 12 |

Consider first the conditions at $t_1$. The total available memory is 24 frames to be distributed among four active processes $P_1$, $P_2$, $P_3$, and $P_4$ whose values of $p_i$ and $w_i(t_i)$ are given in the 4th and 5th rows of the table. The values of $\tilde{m}_i$, $\hat{m}$, r, $m_i$ and $\eta$ are computed as described above, leading to the nonfragmented apportionment of the 24 memory frames among the active processors as indicated in the last row of the table, labeled $\bar{m}_i$. At time $t_2$, the same processes are still active, with new values for $w_i(t_2)$, the working-set size. The updated reallocation of the 24 frames is shown in the $\bar{m}_i$ row. At time $t_3$, processes $P_3$ and $P_4$ are no longer active and are replaced by processes $P_4$ and $P_5$ with an increase of the total available memory to 26 frames. At time $t_4$, process $P_1$ is no longer active, and the 26 frames are reapportioned among the three remaining processes.

## Appendix: Proof of Theorem

(1) Assume that for the given $i \in I$, the condition in (23) holds. We will show that $\hat{m}_i = p_i$. Consider the specific set $m^* = \{m_j^*\}$ with $j \in I$, whose elements $m_j^*$ are defined as follows:

$$m_i^* = p_i \tag{37}$$

$$m_j^* = w_j + \theta(p_j - w_j) \qquad \text{all } j \neq i \tag{38}$$

where $\theta$ is defined as

$$\theta = (M - \Sigma_{j \neq i} w_j - p_i)/(\Sigma_{j \neq i} p_j - \Sigma_{j \neq i} w_j) \; . \tag{39}$$

It follows from (23) that the numerator of the fraction in (39) is nonnegative. This implies $\theta \geq 0$ since the denominator is also nonnegative (see (22)). We now show that $\theta \leq 1$:

$$\Sigma_{j \neq i} p_j - \Sigma_{j \neq i} w_j = \Sigma_{j \in I} p_j - p_i - \Sigma_{j \neq i} w_i \geq M - \Sigma_{j \neq i} w_j - p_i \tag{40}$$

where the inequality in (40) follows from (12). Thus $\theta \leq 1$. From (37) and (38) and the fact that $0 \leq \theta \leq 1$, it follows that

$$w_j \leq m_j^* \leq p_j \qquad \text{for all } j \in I \; . \tag{41}$$

Furthermore, evaluating the sum $\Sigma m_j^*$ and using the expressions in (37), (38), and (39), we obtain

$$\Sigma_{j \in I} m_j^* = p_i + \Sigma_{j \neq 1} m_j^* = p_i + \Sigma_{j \neq 1} w_j + \theta(\Sigma_{j \neq 1} p_j - \Sigma_{j \neq 1} w_j) = M \; . \tag{42}$$

The conditions in (41) and (42) imply that $m^*$ satisfies all the conditions for being a member of the set C, defined in (17), except possibly for the fact that $m_j^*$ as defined in (38) might not be an integer. Let $\{\bar{m}_j^*\}$ be the set of integers obtained by rounding the values of $\{m_j^*\}$ using the procedure described in (32)-(36). Note that

$$\Sigma_{j \in I} \bar{m}_j^* = M \quad , \quad w_j \leq \bar{m}_j^* \leq p_j \tag{43}$$

where the first statement in (43) follows from the fact that the rounding procedure preserves the sum of the original numbers, and the second statement follows from (41). Thus $\bar{m}^* \in C$. Recalling the relationship in (22) which must apply for all $m \in C$, we have

$$w_i \leq \hat{m}_i \leq \bar{m}_i^* \leq \hat{m}_i \leq p_i \tag{44}$$

and noting from (37) that $m_i^* = p_i$, it follows that $\hat{m}_i = p_i$, which is the required result.

(2) Assume that for the given $i \in I$, the condition in (24) holds. Consider the specific set $m^* = \{m_j\}$ whose elements $m_j^*$ are defined as follows:

$$m_i^* = M - \Sigma_{j \neq i} w_j \tag{45}$$

$$m_j^* = w_j \quad , \text{all } j \neq i \, . \tag{46}$$

Note that $m_j^*$ for all j are integer-valued, and that

$$\Sigma_{j \in I} m_j^* = \Sigma_{j \neq i} w_j + M - \Sigma_{j \neq i} w_j = M \, . \tag{47}$$

Note also (45) and (24) imply that $m_i^* < p_i$ while (45) and (12) imply

$$m_i^* = M - \Sigma_{j \in I} w_j + w_i \geq w_i \, . \tag{48}$$

Thus $w_i \leq m_i^* \leq p_i$, and (46) implies $w_j \leq m_j^* \leq p_i$ for all $j \neq i$; hence

$$w_j \leq m_j^* < p_j \qquad \text{all } j \in I \, . \tag{49}$$

Thus (47) and (49) imply that $m^* \in C$ as defined in (17). We now show that the $m_i^*$ given in (45) is the largest value among the $i^{th}$ elements of all $m \in C$:

$$m_i = M - \Sigma_{j \neq i} m_j \leq M - \Sigma_{j \neq 1} w_j = m_i^* \quad , \text{for all } m \in C \tag{50}$$

where the inequality in (50) follows from the fact that $m_j \geq w_j$. Thus, we obtain the required result

$$\hat{m}_i \equiv \max_{m \in C} m_i = M - \Sigma_{j \neq i} w_j < p_i$$

which follows from (49) and (50).

(3) Assume that for the given $i \in I$, the condition in (25) holds. We will show that $\tilde{m}_i = w_i$. Consider a specific set $m^* = \{m_j^*\}$ with $j \in I$, whose elements $m_j$ are defined as follows:

$$m_i^* = w_i \tag{51}$$

$$m_j^* = p_j - \lambda(p_j - w_j) \qquad , \text{all } j \neq i \tag{52}$$

where $\lambda$ is defined as

$$\lambda = (\Sigma_{j \neq i} p_j - M + w_i)/(\Sigma_{j \neq i} p_j - \Sigma_{j \neq i} w_j) \, . \tag{53}$$

It follows from (25) that the numerator of the fraction in (53) is nonnegative. This implies $\lambda \geq 0$ since the denominator is also nonnegative. We now show that $\lambda \leq 1$:

$$\Sigma_{j\neq i}p_j - \Sigma_{j\neq i}w_j = \Sigma_{j\neq i}p_j - \Sigma_{j\in I}w_j + w_i \geq \Sigma_{j\neq i}p_j - M + w_i \tag{54}$$

where the inequality in (54) follows from (12). Thus $\lambda \leq 1$. From (51) and (52) and the fact that $0 \leq \lambda \leq 1$, it follows that

$$w_j \leq m_j^* \leq p_j \qquad \text{for all } j \in I\,. \tag{55}$$

Furthermore, evaluating the sum $\Sigma m_j^*$ and using the expressions in (51), (52), and (53), we obtain

$$\Sigma_{j\in I}m_j^* = w_i + \Sigma_{j\neq 1}m_j^* = w_i + \Sigma_{j\neq 1}p_j - \lambda(\Sigma_{j\neq 1}p_j - \Sigma_{j\neq 1}w_j = M\,. \tag{56}$$

Let $\bar{m}^* = \{\bar{m}_j^*\}$ be the set of integers obtained by rounding the values of $\{m_j^*\}$ using the procedure described in (32)-(36). Evidently, we can use the same arguments of section (1) of the proof above to show that $\bar{m}^* \in C$. Recalling the relationship in (22) which must apply for all $m \in C$, we have

$$w_i \leq \check{m}_i \leq m_i^* \leq \hat{m}_i \leq p_i \tag{57}$$

and noting from (51) that $m_i^* = w_i$, it follows that $\check{m}_i = w_i$, which is the required result.

(4) Assume that for the given $i \in I$, the condition in (26) holds. Consider a specific set $m^* = \{m_j^*\}$ whose elements $m_j^*$ are defined as follows:

$$m_i^* = M - \Sigma_{j\neq i}p_j \tag{58}$$

$$m_j^* = p_j \qquad \text{all } j \neq i\,. \tag{59}$$

Note that $m_j^*$ for all j are integer-valued, and that

$$\Sigma_{j\in I}m_j^* = \Sigma_{j\neq i}p_j + M - \Sigma_{j\neq i}p_j = M\,. \tag{60}$$

Note also (58) and (26) imply that $m_i > w_i$ while (58) and (12) imply

$$m_i^* = M - \Sigma_{j\in I}p_j + p_i \leq p_i\,. \tag{61}$$

Thus $w_i \leq m_i^* \leq p_i$, and (59) implies $w_j \leq m_j^* \leq p_i$ for all $j \neq i$; hence

$$w_j < m_j^* \leq p_j \quad \text{all } j \in I . \tag{62}$$

Thus (60) and (62) imply that $m^* \in C$ as defined in (17). We now show that $m_i^*$ given in (58) is the smallest value among the $i^{th}$ elements of all $m \in C$:

$$m_i = M - \Sigma_{j \neq i} m_j \geq M - \Sigma_{j \neq i} p_j = m_i^* \quad \text{, for all } m \in C \tag{63}$$

where the inequality in (63) follows from the fact that $m_j \leq p_j$. Thus, we obtain the required result

$$\tilde{m}_i = \min_{m \in C} m_i = M - \Sigma_{j \neq i} p_j > w_i \tag{64}$$

which follows from (62) and (63).

This completes the proof of the Theorem.

## References


[1] DENNING, P.J. Working sets past and present, IEEE Trans SE, vol. SE-6, no. 1, January 1980, 64-84.

[2] DENNING, P.J. The working set model for program behavior. Comm. ACM 11, no. 5, May 1988, 323-333.

[3] BIC, L., and SHAW, A.C. The Logical Design of Operating Systems, second ed. Prentice-Hall, 1988, 226-235.

[4] HWANG, K., and BRIGGS, F.A. Computer Architecture and Parallel Processing. McGraw-Hill, 1984, 65-80.