Performance analysis of priority-driven algorithm for multiprocessor model with independent memories: mean flow time criterion

C D Spyropoulos and D J Evans

An analysis is made of the performance of priority-driven algorithms on a homogeneous multiprocessor model with independent memories, when the mean flow time is the performance criterion. The findings support the conjecture that a deterministic analysis could be used in addition to a probabilistic one for systems where the expected performance is more meaningful than the extreme.

Keywords: multiprocessing systems, performance evaluation, algorithms

The priority-driven (PD) algorithm has been analysed under various preset ordering procedures for a homogeneous multiprocessor system, where the completion time is used as the performance criterion. In the present paper, the behaviour of the PD algorithm and the two-phase PD algorithm is examined with regard to mean flow time, under the same preset ordering rules and multiprocessor computing model, to give a view of the behaviour of this algorithm such as that in Reference 4. Optimal scheduling algorithms with respect to the mean flow time criterion have been found for the classical multiprocessor computing model (by which is meant a model that does not have any resource constraints) with identical processors, uniform processors and nonidentical processors. For the model under investigation, the shortest processing time (SPT) ordering rule, discussed in Reference 5, does not produce schedules with optimal mean flow time, and optimal schedules can be obtained only from a specially modified Bruno's algorithm.

The homogeneous multiprocessor model with independent memories consists of m identical, independent, abstract processors \( P = \{P_1, P_2, \ldots, P_m\} \). A fixed size private memory denoted by \( |P_i| \), \( 1 \leq i \leq m \), is associated with the \( i \)th processor. For convenience, the processors are arranged such that \( |P_i| > |P_{i+1}|, 1 \leq i \leq (m - 1) \). On the other hand, the task system consists of \( n \) independent jobs \( J = \{j_1, j_2, \ldots, j_n\} \) where each job is defined by its memory and time requirements. So, the task system can be represented by the three tuple \( (J, |m|, |t|) \).

Let the label \( l_j, 1 \leq j \leq n, \) be the number of processors on which the \( j \)th job can be executed according to the memory constraint.

The PD algorithm is a demand-scheduling algorithm (i.e. one that never intentionally leaves a processor idle), with the following philosophy. When a processor becomes available, the priority list (a permutation of the jobs in the task system) is scanned from left to right, and the first executable job is allocated to that processor. However, the jobs are assumed to be simultaneously available for (nonpreemptive) execution at the beginning of the

8 Baiardi, F, Ricci, L and Vanneschi, M 'Static checking of interprocess communication in ECSP' ACM Sigplan Symp. on Compiler Construction, Montreal, Canada (June 1984)


11 Briatico, D, Ciuffoletti, A, Simoncini, L and Strigini, L 'Error detection/fault treatment in the MuTEAM system' Internal Rep. IEL-CNR (November 1983)


14 Briatico, D, Ciuffoletti, A and Simoncini, L 'A distributed domino-effect free recovery algorithm' Proc. 4th Symp. on Reliability in Distributed Software and Database Systems, Silver Spring, USA (15–17 October 1984)

15 Ciuffoletti, A 'Error recovery in systems of communicating processes' Proc. 7th Int. Conf. on Software Engineering, Orlando, FL, USA (March 1984) pp 6–17
schedule, and, when two or more processors are available simultaneously, the first executable job is assigned to the largest indexed processor. The preset ordering procedures used to form a priority list are: arbitrary (Rand), largest memory first (LMF) (i.e. job \( j_k \) precedes \( j_k \) if \( t_j < t_k \)), shortest processing time (i.e. job \( j_j \) precedes job \( j_k \) if \( t_j < t_k \)), largest processing time (LPT) (i.e. job \( j_j \) precedes job \( j_k \) if \( t_j > t_k \)), largest memory, shortest time (LMS) (i.e. job \( j_j \) precedes job \( j_k \) if \( a_j < a_k \) or \( t_j < t_k \)) or largest memory, largest time (LMLT) (i.e. job \( j_j \) precedes job \( j_k \) if \( a_j > a_k \) or \( t_j > t_k \)) ordering. The two-phase PD (PD') algorithm initially performs as a normal PD algorithm, and the assigned jobs at each processor are then sorted in nondecreasing order, according to their processing times.

It is well known that the mean flow time \( \bar{O} \) of a given schedule is the sum of the finishing times of all the jobs, divided by the number of jobs in the task system, i.e.

\[
\bar{O} = \frac{1}{n} \sum_{j=1}^{n} f_j
\]

where \( f_j \) is the finishing time of job \( j \), or

\[
\bar{O} = \frac{1}{n} \sum_{j=1}^{n} f_j
\]

where \( f_j \) is one greater than the number of jobs following \( j \) on the \( r \)th processor to which \( j \) is assigned.

A scheduling algorithm is evaluated by its worst-case or its expected behaviour. The former is estimated using a deterministic approach, while the latter is estimated using a probabilistic one. The deterministic approach is a suitable tool for systems in which a guaranteed level of performance is required, while the probabilistic one is useful for systems in which the expected performance is needed. However, it is debatable whether the deterministic approach could also be useful for systems where the expected performance is more meaningful. This could happen where the worst-case behaviour of an algorithm agrees with its expected behaviour.

**PRELIMINARIES**

Before the scheduling algorithms are analysed, it is necessary to define a number of variables, to declare the sets of jobs, and to present some lemmas, which are used in the following section.

\( \bar{F} \) is defined as the set of jobs with \( \bar{F} = \{j_i \mid 1 \leq i \leq m \} \)

\( \bar{J}_j \) is the set of jobs being scheduled on \( \bar{F} \), \( 1 \leq i \leq m \)

\( \bar{C}_r \) is the corresponding number of jobs belonging to \( \bar{C}_r \).

However, the proof of the theorems in the next section is based on the following lemmas, which are given here without proof (see Reference 10 for proofs).

**Lemma 1**

If \( a_{x+1} \geq a_{x+2} \geq \ldots \geq a_{x+y} \) for \( j = 1, q \mid d \) and \( q, d \geq 1 \) and \( q, d \in Z^+ \)

\[
\left( k+s+1 \right) a_{s+1} + \left( k+s+2 \right) a_{s+2} + \ldots + \left( k+s+d \right) a_{s+d} \]

\[
\frac{q(\bar{s}+1) + a_{s+2} + \ldots + a_{s+d}}{q(\bar{s}+1) + a_{s+2} + \ldots + a_{s+d}}
\]

\[
\leq \frac{k}{q} + d - \frac{d-1}{2q}
\]

where \( k \geq 0 \) and \( k \in Z^+ \).

(NB: This lemma generalizes the one proved by Stewart (page 4 of Reference 11) who proved the above inequality for \( k = 0 \) and \( s = 0 \).)

**Lemma 2**

If \( a_1, a_2, \ldots, a_n \in R^+ \), \( n \geq 2 \) and

\[
\sum_{i=1}^{n} a_i \leq f_q(c)
\]

where \( s, q, d, \nu \in Z^+ \),

\[
q = \sum_{i=1}^{n} d_i, q = 1(1)\nu, \nu \geq 2
\]

\[
\sum_{i=1}^{n} d_i = n
\]

\[
c_{s+1} \in R^+, c_{s+1} \neq c_{s+1+1} \text{ for } 1 \leq s \leq d_q - 1, \text{ and } d_q \text{ is a positive function, then}
\]

\[
\sum_{i=1}^{\nu} \sum_{j=1}^{q} \left( c_{s+j} a_{s+j} \right) \leq \max \left\{ f_q(c) \right\}
\]

**Lemma 3**

Let a task system \( r \} \) with \( n \) independent jobs be scheduled on a homogeneous multiprocessor system with independent memories. If \( \bar{O}_{\text{OPT}} \) corresponds to the optimal mean flow time of the above task system, then

\[
\bar{O}_{\text{OPT}} \geq t_1 + t_2 + \ldots + t_n
\]

**Lemma 4**

Let a task system \( r \} \) with \( n \) independent jobs, which are in an SPT ordering (i.e. \( t_1 \leq t_2 \leq \ldots \leq t_n \)), be scheduled on a homogeneous multiprocessor system with \( m \) independent memories. Then

\[
\bar{O}_{\text{OPT}} \geq t_1 + t_2 + \ldots + t_m + 2(t_{m+1} + \ldots + t_{2m})
\]

\[
\ldots + \frac{n}{m} \left( t_{\frac{n}{m}} \right)^{m+1} + \ldots + \left( t_{\frac{n}{m}} \right) \]

\( *R^+ \) represents the set of real positive numbers, \( *Z^+ \) represents the set of positive integers

\( 1^i = \{ (i) \} \) means that it takes all the values from \( i_1 \) to \( i_2 \) with a step of \( i_3 \)

\( 1[a] \) is the least integer greater than or equal to \( a \), and \( 1[a] \) is the greatest integer less than or equal to \( a \)
where \( \left\lceil \frac{n}{m} \right\rceil \) is the first integer greater than or equal to \( n \) which can be divided by \( m \), and the tasks \( J_{n+1}, \ldots, J_m \) have zero processing time requirements.

**Lemma 5**

Let a task system \( (J, \{m_j\}, \{t_{ij}\}) \) with \( n \) independent jobs, which are in an LMST ordering (i.e., \( t_{ni} \leq t_{n-1,i} \leq \ldots \leq t_i \leq t_{1,i} \leq i \leq m \)), be scheduled on a homogeneous multiprocessor system with \( m \) independent memories. Then

\[
\bar{\omega}_{\text{OPT}} \geq \left( t_1 + 2t_2 + \ldots + t_{n-1} \right) + \left( \left( 2t_1^2 + 2t_2^2 + \ldots + 2t_{n-1}^2 \right) \right)
\]

\[
+ 2\left( \frac{2t_1^3}{3} + \frac{2t_2^3}{3} + \ldots + \frac{2t_{n-1}^3}{3} \right) + \ldots + \left( \left( \frac{2t_1^m}{m+1} + \frac{2t_2^m}{m+1} + \ldots + \frac{2t_{n-1}^m}{m+1} \right) \right)
\]

\[
+ \ldots + \left( \left( \frac{2t_1^m}{m+1} + \frac{2t_2^m}{m+1} + \ldots + \frac{2t_{n-1}^m}{m+1} \right) \right)
\]

where \( t_i \) are the time requirements of jobs

\[ J_{j_i} \in F_i, 1 \leq j \leq n_i, 1 \leq i \leq m, \quad \sum_{i=1}^{m} n_i = n \]

where \( \left\lceil \frac{n_i}{i} \right\rceil \) is the first integer greater than or equal to \( n_i \) which can be divided by \( i \), and the jobs \( J_{n+1}, \ldots, J_m \), have zero processing time requirements.

**WORST-CASE ANALYSIS**

Let \( \bar{\omega}_{\text{PD}} \) be the mean flow time of the schedule constructed by the PD algorithm, when the priority list is formed by a heuristic ordering procedure, and let \( \bar{\omega}_{\text{OPT}} \) be the optimal mean flow time for a given task system \( (J, \{m_j\}, \{t_{ij}\}), j = 1(1)n \).

**Theorem 1**

Let the priority list be in an arbitrary (Rand), LMF, LPT or LMLT ordering. Then

\[
\frac{\bar{\omega}_{\text{PD}}}{\bar{\omega}_{\text{OPT}}} \leq n'_{\text{max}}
\]

where \( n'_{\text{max}} = \max_{1 \leq i \leq m} \{ n_i \} \).

**Proof**

Let \( \bar{\omega}_{\text{PD}}, \bar{\omega}_{\text{PD}}, \ldots, \bar{\omega}_{\text{PD}} \) be the contribution to the mean flow time of jobs \( J_i \in C \), with time requirements \( t_i, j = 1(1)n_i, i = 1(1)m \) respectively, so

\[
\bar{\omega}_{\text{PD}} = \bar{\omega}_{\text{PD}} + \bar{\omega}_{\text{PD}} + \ldots + \bar{\omega}_{\text{PD}}
\]

From the definition of mean flow time given by equation (1), the contribution of the jobs scheduled on the \( P_i \) processor is:

\[
\bar{\omega}_{\text{PD}} = n_i t_1^n + (n_i - 1) t_2^n + \ldots + 2t_{n_i-1} + t_{n_i}^n, 1 \leq i \leq m
\]

Therefore, because of equation (2),

\[
\bar{\omega}_{\text{PD}} = \sum_{i=1}^{m} (n_i t_1^n + (n_i - 1) t_2^n + \ldots + 2t_{n_i-1} + t_{n_i}^n)
\]

From Lemma 3,

\[
\bar{\omega}_{\text{OPT}} \geq \sum_{i=1}^{m} (t_1^n + t_2^n + \ldots + t_{n_i}^n)
\]

since

\[ n = \sum_{i=1}^{m} n_i \]

Because \( t_i \in R^+ \) for \( j = 1(1)n_i \) and \( i = 1(1)m \), from equations (3) and (4)

\[
\frac{\bar{\omega}_{\text{PD}}}{\bar{\omega}_{\text{OPT}}} \leq \sum_{i=1}^{m} \sum_{j=1}^{n_i} \left( \frac{(n_i - 1) t_j^n}{t_1^n + t_2^n + \ldots + t_{n_i}^n} \right)
\]

Clearly

\[
\frac{n_i t_1^n + (n_i - 1) t_2^n + \ldots + 2t_{n_i-1} + t_{n_i}^n}{t_1^n + t_2^n + \ldots + t_{n_i}^n}
\]

\[ \leq n_i \quad \text{for } i = 1(1)m \]

Applying Lemma 2 for the inequality (6) where

\[ v = m, q = i, d_a = n_i, s = \sum_{i=1}^{1} n_i \]

\[ c_{s+j} = n_i + 1 - j, d_{s+j} = t_j, j = 1(1)n_i \]

and \( i = 1(1)m \), the following inequality is obtained:

\[
\sum_{i=1}^{m} \sum_{j=1}^{n_i} \left( \frac{(n_i - 1) t_j^n}{t_1^n + t_2^n + \ldots + t_{n_i}^n} \right) \leq \max_{1 \leq i \leq m} \{ n_i \} \]
From inequalities (5) and (7), the following result is obtained:

$$\frac{-\bar{\omega}_{PD}}{\bar{\omega}_{OPT}} \leq \max_{1 \leq i \leq m} \{n_i \} = n_{\text{max}}^*$$

which proves the theorem. (Equality in Theorem 1 holds only if $n_{\text{max}}^* = 1$.)

Where the priority list is formed by an LMF, LPT or LMLT ordering procedure, inequalities (2) and (4) cannot be improved. Therefore, these preset ordering rules can offer no improvement in a worst-case sense over an arbitrary ordering for the PD algorithm, when the performance criterion is the mean flow time.

Moreover, the following example shows the proven bound can be approached asymptotically.

**Example 1**

Let the task system $(J_i, |m|, |t_i|)$ be defined by:

$$f_{i+j+1}: (P_i+1, X), \quad 0 \leq i \leq m-1$$

$$f_{i+j+1}: (P_i+1, e), \quad 0 \leq i \leq m-1, 2 \leq j \leq r$$

where $n = mr, n_i = r$ for $i = 1(1)m, \{e, X\} \in R^+$ and $e < X$.

Clearly, the priority list $L_1 = (J_1, J_2, \ldots, J_n)$ is in LMF or LMLT order; while the priority list $L_2 = (J_1, J_{m+1}, J_{m+2}, \ldots, J_{m-r+1}, J_{m-r+2}, \ldots, J_n, J_{r+1}, J_{r+2}, \ldots, J_{m-1}, J_{m})$ in LTF order. Either of these two lists can be chosen as a Rand ordering. The schedule resulting from the priority list $L_1$ or $L_2$, when the PD algorithm is used, is shown in Figure 1. The corresponding optimal schedule is given in Figure 2.

The ratio of the mean flow times of these two schedules is:

$$\frac{-\bar{\omega}_{PD}}{\bar{\omega}_{OPT}} = \frac{\max_{1 \leq i \leq m} \{r+i\} \leq \epsilon}{m \leq \epsilon + \frac{\max_{1 \leq i \leq m} \{r+i\}}{m \leq \epsilon}}$$

but

$$\lim_{\epsilon \to 0} \frac{-\bar{\omega}_{PD}}{\bar{\omega}_{OPT}} = \frac{m \leq \epsilon + \frac{\max_{1 \leq i \leq m} \{r+i\}}{m \leq \epsilon}}$$

which is the value predicted by Theorem 1.

It can be observed that if the number of processors increases and the number of jobs in the task system remains constant or decreases, then the value of the worst-case performance bound of the PD algorithm under the considered ordering rules decreases, since $n_{\text{max}}^* = \max_{1 \leq i \leq m} \{n_i \}$ is expected to decrease as well. In the reverse situation, the value of the worst-case bound increases. However, small increases or decreases in the number of processors or jobs in the task system might not affect the performance of the algorithm, since $n_{\text{max}}^*$ might not change.

**Theorem 2**

Let the priority be in an SPT ordering. Then

$$\frac{-\bar{\omega}_{PD}}{\bar{\omega}_{OPT}} \leq \min \left\{ \max_{1 \leq i \leq m} \left( \frac{v_i - \frac{(v_j-1)}{n_i}}{2 \frac{n_i}{n_i}} \right) \right\}, \text{ where } v_i = \max_{i \in G_j} \{l_j\}$$

**Proof**

Let $\bar{\omega}_{PD}, \bar{\omega}_{OPT}, \ldots, \bar{\omega}_{OPT}$ be the contribution of the jobs $J_i \in G_i$ to the mean flow time of the schedule constructed by the PD algorithm, while the priority list is formed by the SPT ordering, for $i = 1(1)m$ respectively. The time requirements of the jobs $J_i \in G_i$ are denoted by $t_i$. Also, let $\bar{\omega}_{OPT}, \bar{\omega}_{SPT}, \ldots, \bar{\omega}_{OPT}$ be the contribution of the jobs $J_i \in G_i$ to the optimal mean flow time, for $i = 1(1)m$ respectively. Then, because of equation (1),

$$\bar{\omega}_{PD} = n_i t_{i+n_i} + (n_i - 1) t_{i+n_i-1} + \ldots + t_{i}$$

for $i = 1(1)m$

where

$$t_{i+n_i} \leq t_{i+n_i-1} \leq \ldots \leq t_2 \leq t_{i}.$$  

Applying Lemma 4 to the jobs of the $G_i (1 \leq i \leq m)$ set being scheduled on the first $v_i$ processors, we
obtain
\[ O_x = (t_1' + t_2' + \ldots + t_i') + 2(t_{i+1} + \ldots + t_n') \]
\[ \leq \sum_{j=1}^{n'} \omega_{OPT} \]
where \( t_i = \ldots = t_i' = 0 \) for \( i = 1(1)m \).

From equations (8) and (9) and since \( \omega_{PD} = \sum_{j=1}^{n} \omega_{PD} \), we get
\[ \frac{\omega_{PD}}{\omega_{OPT}} \leq \frac{\sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) \left( (r-1)v_i t_{i+1} \right) }{\sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) t_{i+1} v_j} \]
(10)

Using Lemma 1 with \( k = 0, d = v, q = r, s = (r-1)v_i \) and \( a_{s+t} = t_{s+t} \) \( j = 1(1)m \) for \( r = 1(1) \left( \frac{n_i'}{v_i} \right) \) and applying Lemma 2 on the resulting inequalities where \( q = r, d_q = v, q = 1 \leq i \leq m, v = \left( \frac{n_i'}{v_i} \right), s = (r-1)v_i \), we obtain
\[ \frac{\sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) \left( (r-1)v_i t_{i+1} \right) }{\sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) t_{i+1} v_j} \]
\[ \leq \frac{v_i}{v_j} - \frac{(v_i - 1)}{2} \] for \( i = 1(1)m \)
(11)

Again, applying Lemma 2 for the inequalities (11), where \( q = i \) and \( v = m \), we get
\[ \frac{\sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) \left( (r-1)v_i t_{i+1} \right) }{\sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) t_{i+1} v_j} \]
\[ \leq \max_{1 \leq i \leq m} \left( \frac{v_i - (v_i - 1)}{2 \left( \frac{n_i'}{v_i} \right)} \right) \]

Finally, because of inequality (10), the following is obtained:
\[ \frac{\omega_{PD}}{\omega_{OPT}} \leq \max_{1 \leq i \leq m} \left( \frac{v_i - \frac{(v_i - 1)}{2}}{\left( \frac{n_i'}{v_i} \right)} \right) \]
(12)

Another way to find an upper bound is to use Lemma 3 for the jobs in the set \( C_i, 1 \leq i \leq m \), to bound the optimal mean flow time, instead of Lemma 4. So
\[ \omega_{OPT} \geq t_1' + t_2' + \ldots + t_n', \text{for } j = 1(1)m \]
(13)

and hence, because of equation (8),
\[ \frac{\omega_{PD}}{\omega_{OPT}} \leq \frac{\sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) t_j }{\sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) t_j} \]

On the other hand, applying Lemma 2 for \( v = m \) to the inequalities provided by Lemma 1 for \( k = 0, d = n_i', s = 0, q = 1 \) and \( a_{s+t} = t_{j}, j = 1(1)m \), we obtain
\[ \sum_{i=1}^{m} \sum_{j=1}^{n'} \left( \frac{n_i'}{v_i} \right) \sum_{j=i}^{n'} \left( \frac{v_i}{v_j} \right) t_j \leq \max_{1 \leq i \leq m} \left( \frac{n_i' + 1}{2} \right) \]
(14)

Therefore, because of inequalities (13) and (14), the worst-case bound of the PD algorithm when the priority list is formed by an SPT procedure is
\[ \frac{\omega_{PD}}{\omega_{OPT}} \leq \min_{1 \leq i \leq m} \left( \frac{v_i - \frac{(v_i - 1)}{2}}{\left( \frac{n_i'}{v_i} \right)} \right) \]

The value of the second factor is less than the value of the first if \( n_{max}' = \left( \frac{3m}{2} \right) - 1 \). This is true because
\[ \left( m - \frac{(m-1)}{2} \right) \geq \left( \frac{n_i' + 1}{2} \right) \]
only if \( n_i' \leq \left( \frac{3m}{2} \right) - 1 \). The last statement can be verified by an induction process. (Equality in Theorem 2 holds only if \( n_{max} = 1 \).

This theorem indicates that, for the SPT ordering procedure, although the optimal mean flow time cannot be produced as for the classical homogeneous multiprocessor model, an improvement is gained over the extreme performance of the PD algorithm under an arbitrary ordering. It can be observed that, the greater the value of \( n_{max} \), the larger the difference between the guaranteed performance levels of the arbitrary and SPT...
preset ordering rules. Finally, the worst-case bound is expected to increase as the number of processors increases.

**Theorem 3**

Let the priority list be in an LMST ordering. Then

\[
\bar{\omega}_{PD} \leq \min \left( \max_{1 \leq i \leq m-1} \left( \max_{1 \leq i \leq r} \left( \frac{k + \frac{i-1}{2}}{r} \right) \right) \right)
\]

\[
\left( m - \frac{(m-1)}{2} \right), \max_{1 \leq i \leq m-1} \left( \frac{n_j + 1}{2} \right)
\]

where \( k \) is the maximum number of the jobs with \( i > j \) that have been scheduled on the \( i \)-th processor, \( 1 \leq i \leq m-1 \) (i.e., \( k = \max_{1 \leq i \leq m-1} \left( n_i - n_j \right) \)).

**Proof**

Let \( \bar{\omega}_{PD}^1, \bar{\omega}_{PD}^2, \ldots , \bar{\omega}_{PD}^m \) be the contribution of jobs \( j \in F_i \) to the mean flow time, for \( i = 1(1)m \) respectively, when the jobs in the priority list are in an LMST ordering (i.e., \( t_{1i} \leq t_{2i} \leq \cdots \leq t_{ni} \leq t_{ki} \)), where \( t_{ji} \) are the time requirements of the jobs \( j \in F_i \), \( j = 1(n_i) \), and \( i = 1(1)m \). Also, let \( \bar{\omega}_{OPT}^1, \bar{\omega}_{OPT}^2, \ldots , \bar{\omega}_{OPT}^m \) be the contribution of the jobs in the set \( F_i \) to the optimal mean flow time, for \( i = 1(1)m \) respectively.

Applying equation (1) and bearing in mind the PD algorithm, the contribution of the jobs belonging to \( F_i \) (i.e., those with \( j = i \)) is always

\[
\bar{\omega}_{PD}^i \leq (k+1) t_{1i} + (k+2) t_{2i} + \ldots + (k+n_i) t_{ni}
\]

for \( i = 1(1)(m-1) \)

(15)

since the jobs with \( j > i \) will be scheduled on \( P_i \) after the jobs \( j \), with \( j = i \), having been assigned to that processor. However, when \( i = m \),

\[
\bar{\omega}_{PD}^m \leq t_{1m}^m + 2 t_{2m}^m + \ldots + n_m t_{nm}^m
\]

(16)

since only the jobs \( j \in F_m \) can be scheduled on \( P_m \). Because

\[
\bar{\omega}_{PD} = \sum_{i=1}^{m} \bar{\omega}_{PD}^i, t_j^i \in R^*
\]

and because of inequalities (15) and (16),

\[
\bar{\omega}_{PD} \leq \sum_{j=1}^{m-1} \left[ (k+1) t_{1j} + (k+2) t_{2j} + \ldots + (k+n_j) t_{nj} \right] + \sum_{j=1}^{m} \left[ \frac{2 n_j^m}{m} \right]
\]

(17)

From Lemma 5, \( \bar{\omega}_{OPT} \) is bounded by

\[
\bar{\omega}_{OPT} \geq (r_1 + r_2 + \ldots + n_1 r_1) + \sum_{j=1}^{m} \left[ \frac{2 n_j^m}{m} \right]
\]

(18)

where \( t_{n_j+1} = \ldots = t_{i} = 0 \) for \( i = 1(1)m \). Therefore, from inequalities (17) and (18),

\[
\bar{\omega}_{PD} \leq \sum_{j=1}^{m-1} \left[ (k+1) t_{1j} + (k+2) t_{2j} + \ldots + (k+n_j) t_{nj} \right] + \sum_{j=1}^{m} \left[ \frac{2 n_j^m}{m} \right]
\]

vol 1 no 1 october 1985
the results obtained for the PD and PD* algorithms respectively, when the memory and time requirements were selected from the uniform distribution. The detail simulation results are presented in Reference 10.

Many conclusions can be drawn from the results of Figures 5 and 6. First of all, as was expected, the average performance of the algorithms under any priority list is much better than the corresponding worst-case performance. Also, as the number of processors increases, the average performance of the PD algorithm with the Rand, LMF, LPT and LMST orderings decreases. With the SPT and LMST orderings, however, it increases and the distances between the performance curves of the

![Figure 4. Biased distribution favouring (a) small memory requirements and (b) large memory requirements](image)

![Figure 3. Biased distribution favouring: (a) short-time requirements and (b) long-time requirements](image)

![Figure 5. Test 1 for PD algorithm; (a) Rand, (b) LMF, (c) SPT, (d) LPT, (e) LMST, (f) LMLT](image)

![Figure 6. Test 1 for PD* algorithm; (a) Rand, (b) LMF, (c) SPT, (d) LPT, (e) LMST, (f) LMLT](image)
ordering rules become shorter as the number of
processors increases. The utilization of memory in
contributing to the SPT ordering does not offer a great
improvement in the average performance. The PD* algorithms
do offer considerable improvement on
average performance over the PD algorithms only
when the priority list is in a Rand, LMF, LPT or LMLT
ordering, while, for the other ordering rules, there is
very little or no improvement. Also, it should be noted
that the ordering procedures with identical worst-case
bounds may have significantly different expected
performances (see Figure 5 for the Rand, LMF, LPT and
LMLT orderings).

Further ordering rules with distinct, though close,
bounds may have expected performances that are
indistinguishable by simulation techniques (see
Figures 5 and 6). Moreover, the PD algorithms under
SPT and LMST orderings display very good perfor-
mance characteristics. Their maximum difference from
the optimal performance is approximately 5%. In some
cases, two ordering rules are ranked even when their
performances differ less than 0.05. This is because an
experiment proceeds until all the confidence intervals
of the statistics for each ordering rule and algorithm
become less than 0.05. Hence, some of the ordering
rules produce confidence intervals less than 0.05.

From the results of the other experiments that
appear in Reference 10, it is seen that the average
performance is better or worse depending on the
choice of the biased distributions for the memory and
time requirements. Such results are well predicted
from the proven worst-case bounds. However, there is
perfect agreement between the ranking of the extreme
and average performances of the PD and PD* algo-
rithms under the examined preset ordering rules,
respectively.

CONCLUDING REMARKS
Although the proven worst-case bounds are not
impressive, they provide a thorough understanding of
the behaviour of the considered algorithms under the
various preset ordering procedures and in different
situations. However, the high degree of correlation
between the performance behaviour of the algorithm
predicted by the deterministic approach and estima-
ted by the probabilistic approach using simulation
techniques supports the conjecture that the deter-
mindistic analysis could supplement simulation studies
for systems where the expected performance is more
meaningful than the extreme. In reality, the deter-
nistic analysis becomes a supplementary and not a
superfluous tool, since, from the simulation studies:

- The rate of increase or decrease of the performance
  of the algorithms as the system parameters change
  values would not be known.
- When, for some operational environments, an
  algorithm's ranking changes, its behaviour could not
  be explained properly.
- The chosen configurations of the model, as a sample
  to be simulated, might not be sufficient.

From the present analysis and the work presented in
References 1, 2 and 3, it can be seen that the PD*
algorithm under the LMULT ordering procedure is the
best to satisfy both performance criteria reasonably
well for any operational environment. On the other
hand, the PD algorithm under the SPT ordering rule is
the optimum strategy to obtain all-round performance
for any operational environment, when the mean flow
time is the performance criterion.

Further research could be done on the same model
by examining the QAD algorithms introduced by
Clark11, which promise a good turnaround perfor-
mance, and by investigating their performance under
both criteria (i.e., completion and mean flow time).
Also, an upper worst-case bound for Bruno's algorithm,
when the completion time is considered as the
performance criterion, is an interesting point to be
pursued.

Finally, it is concluded that the support of the
considered conjecture was based on the proven
informative worst-case bounds. So the researcher of
the deterministic approach is advised to include in his
analysis as many critical parameters that affect the
performance of the algorithm as possible in order to
obtain more meaningful results.

REFERENCES
1 Kafura, D. G and Shen, V. Y. "Task scheduling on a
multiprocessor system with independent
memories." SIAM J. Comput. Vol 6 No 1
2 Spyropoulos, C. D and Evans, D. J. "Generalised
worst-case bounds for a homogeneous multi-
processor model with independent memories —
completion time performance criterion." Perf.
3 Spyropoulos, C. D and Evans, D. J. "Worst-case
bounds with respect to completion time per-
formance criterion for a homogeneous multi-
processor computing model." Technical Report No
194. Dept. of Computer Studies, LUT, UK
(August 1983).
4 Coffman, E. G. Jr and Sethi, R. "Algorithms minimising
mean flow time — schedule length properties.
5 Conway, R. W., Maxwell, W. L. and Miller, L. W. "Theory
of scheduling." Addison-Wesley, USA (1967).
6 Horowitz, E. and Sahni, S. "Exact and approximate
algorithms for scheduling nonidentical processors.
pp 317–327.
7 Bruno, J., Coffman, E. G. Jr and Sethi, R. "Algorithms
for minimising mean flow time." Proc. IFIPS
8 Bruno, J. "A scheduling algorithm for minimising
mean flow time." Technical Report No 141,
Computer Science Dept., Pennsylvania State
University, USA (1973).
9 Coffman, E. G. Jr (Ed) "Computer and job-shop
scheduling theory." Wiley, USA (1976) p 44.
10 Spyropoulos, C D and Evans, D J Performance analysis of priority-driven algorithms with respect to mean flow time criterion for a homogeneous multiprocessor model Technical Report No 198, Dept. of Computer Studies, LUT, UK (September 1983)

11 Clark, D Scheduling independent tasks as non-identical parallel machines to minimise mean flow time Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, USA (June 1974)
