Minimizing the Mean and Maximum Finishing Time
on Uniform Processors†

Teofilo Gonzalez, November 1978
CS-78-22

Computer Science Department
The Pennsylvania State University
University Park, Pennsylvania 16802

Abstract

The problem of preemptively scheduling a set of $n$ independent tasks on $m$ uniform processors is discussed. An algorithm to obtain preemptive schedules with bounded maximum finish time and minimum mean finishing time is presented. The algorithm is of time complexity $O(nm)$ and introduces no more than $O(nm)$ preemptions.

keywords: uniform processors, preemptive schedules, OFT, OMFT, β:OMFT, OFT:OMFT, polynomial complexity.

†This research was supported in part by NSF grant MCS 77-21092.

## I. Introduction

There are $n \geq 1$ independent tasks to be scheduled on a uniform processor system. Tasks shall be denoted by $\tau_1, \tau_2, \ldots, \tau_n$ and have execution time requirements $t_1 \leq t_2 \leq \ldots \leq t_n$. A uniform processor system consists of $m \geq 1$ processors, denoted by $P_1, P_2, \ldots, P_m$ with relative speeds $s_1 \geq s_2 \geq \ldots \geq s_m$. An identical processor system is a special case of the former, when $s_i = s_{i+1}$ for $1 \leq i < m$.

Let $w_i > 0$ be the weight given to task $\tau_i$ and $f_i$ be its completion time in schedule S ($f'_i$ in schedule S'). The weighted mean finishing time (wmft) for schedule S is $\Sigma w_i f_i / n$. An optimal weighted mean finishing time schedule (OWMFT) is one with the least wmft. The weights are said to be agreeable when $w_1 \geq w_2 \geq \ldots \geq w_n$. For the case when $w_i = w_{i+1}$, $1 \leq i < n$, these definitions are denoted mean finishing time (mft) and optimal mean finishing time schedules (OMFT). The finish time (ft) for schedule S is the $\max\{f_i\}$. An optimal finish time schedule (OFT) is one with the least finish time. An OFT:OMFT schedule is one with the least mft from the set of all possible OFT schedules. A β:OMFT schedule is one with the least mft from the set of all possible schedules with $ft \leq \beta$.

A preemptive schedule is one in which it is possible to interrupt the execution of a task and resume it at a later time possibly on a different processor. A nonpreemptive schedule is one in which once a task starts execution on some processor, it will continue executing on the same processor without interruption until completion.

Scheduling problems naturally arise in different areas. Examples of applications appear in [CMM] and for the problems studied in this paper see [G2]. Preemptive scheduling has received considerable attention, e.g., [C], [CMM], [G1], [GS1], [GS2], [GS3], [HLS], [LL], [LY], [Mc], [MC1], [MC2], [SG], [U]. Most of these papers present results concerning OFT preemptive schedules. In section II an algorithm for constructing $\beta$:OMFT preemptive schedules is studied. Special cases of this problem are the construction of OFT:OMFT and OMFT preemptive schedules.

For identical processors, McMaughton [Mc] presents a $O(n)$ algorithm to obtain OFT preemptive schedules. The maximum number of preemptions introduced is $m - 1$. These bounds on the time complexity and the maximum number of preemptions are best possible. For uniform processors systems, Liu and Yang [LY] obtained a lower bound for the length of an OFT preemptive schedule. The bound is $\omega \geq \max\{\max_{1\leq j<m}\{{}_{(n-j+1)}T/S_j\}, {}_{(1)}T/S_m\}$, where ${}_{(j)}T = \sum_{j\leq i\leq n} t_i$, $S_j = \sum_{1\leq i\leq j} s_i$ and $n > m$. For the special case $s_i = 1$, $1 \leq i < m$, they also showed that $\omega$ is the length of the OFT preemptive schedule and that an algorithm to construct such a schedule is of polynomial time complexity. Horvath, Lam and Sethi [HLS] present a polynomial time bounded algorithm to construct OFT preemptive schedules. The finish time of such a schedule is $\omega$. Gonzalez and Sahni [GS1] present a $O(n + m)$ algorithm, which introduces at most $2(m - 1)$ preemptions. Both these bounds on the time complexity and the maximum number of preemptions are shown to be best possible.

McNaughton [Mc] shows that any OMFT preemptive schedule for identical processors can be transformed to another schedule with at most the same mft but no preemptions. An OMFT nonpreemptive schedule [CMM, p. 26] for identical processors can be constructed in $O(n \log n)$ time. For uniform processor systems, this is not the case. Lawler and Labetoulle [LL] present an algorithm, based on the solution to a linear programming problem and n open shop problems. The maximum number of preemptions is $O(nm^2)$. As there is no known polynomial time algorithm to solve LP problems, the worst-case time complexity is exponential on the number of tasks and processors. The algorithm presented in section II can be adapted to solve this problem. The time complexity is $O(nm)$ and the maximum number of preemptions introduced is $O(nm)$.

The problem of obtaining OWMFT preemptive schedules for identical processors is NP-hard ($m \geq 2$) [LL].

In [G2], an algorithm to obtain $\beta$:OMFT preemptive schedules for identical machines is presented. The time complexity for the algorithm is $O(nm)$ and introduces $m - 1$ preemptions. For nonpreemptive scheduling, [BCS] present approximation algorithms to minimize the ft and mft. These results are summarized in [C, pp. 42-49].

An algorithm to obtain $\beta$:OMFT preemptive schedules for uniform processor systems is presented in section II. The time complexity is $O(nm)$ and the maximum number of preemptions introduced is $O(nm)$.

## II. β:OMFT Preemptive Schedules for Uniform Processors

An algorithm to obtain OMFT preemptive schedules for uniform processor systems in which each task is required to complete by time β is presented. Note that

$$\beta \geq \max\{ \max_{1 \leq j \leq m} \{ {}_{(n-j+1)}T/S_j \}, {}_{(1)}T/S_m \}$$

as otherwise such a schedule could not be constructed. In case of equality the schedule obtained is an OFT:OMFT preemptive schedule. For β sufficiently large the problem is that of obtaining an OMFT preemptive schedule.

β:OMFT preemptive schedules can be obtained by an algorithm based on the solution to a linear programming problem and n open shop problems. The formulation is as the one in [LL] for OMFT preemptive schedules on uniform machines, but requires the restriction $f_n \leq \beta$. The correctness for this approach follows from lemma 2 and the formulation given by [LL]. Known algorithms for LP problems are in the worst case of exponential time complexity, in this case, exponential on the number of jobs and machines. In addition, too many preemptions are introduced. An algorithm of time complexity O(nm) for this problem is presented and analyzed. The maximum number of preemptions introduced is O(nm). The algorithm is similar to the one in [G1]; however, it is textually more complex.

Algorithm Uβ:OMFT considers task $\tau_i$ at step i. $\tau_i$ is assigned in such a way that its completion time is as early as possible, provided it is late enough so that all remaining tasks can be scheduled to complete no later than β. Initially the completion time (t) for $\tau_i$ is determined.

In order to simplify the bookkeeping operations involved in managing blocks of idle time, a subset of tasks is assigned before $\tau_i$. These tasks have the property that it is not feasible to schedule them to complete before time $\beta$ in any feasible schedule including all previous assignments together with $\tau_i$ scheduled to complete before time $t$. $\tau_i$ is then assigned to complete at time $t$. In order to simplify the assignments in this phase, idle time is partitioned into disjoint blocks (disjoint processors). These blocks are initialized by procedure INITIALIZE and the assignments are made final (in Q) by procedure TERMINATE. Procedure ASSIGN, schedules tasks in sections of two disjoint processors.

Let $M = \{1, 2, \ldots, m\}$ and $N = \{1, 2, \ldots, n\}$. During the execution of the algorithm, processor $P_i$ is busy from time $0$ to $\mu_i$. Processor indices are partitioned into sets $I = \{i_1, i_2, \ldots, i_\ell\}$ and $M - I$. The second set corresponds to those processors made critical (processors for which $\mu$ was set to $\beta$). Initially all processors belong to the first set. Task indices are partitioned into sets $A = \{a_1, a_2, \ldots, a_q\}$ and $N - A$. The first set corresponds to tasks not yet scheduled. $Q_j$ is the schedule for processor $P_j$. $Q_j$ consists of tuples of the form $(i, s, f)$. Tuple $(i, s, f)$ indicates that $\tau_i$ is to be executed from time $s$ to time $f$ by processor $P_j$. Let $f_i$ be the completion time for task $\tau_i$ in schedule Q. Assume $f_i = 0$ for $i \leq 0$.

Before presenting the algorithm, we outline the general strategy. At some point, it is required to schedule tasks $\tau_{a_1}, \tau_{a_2}, \ldots, \tau_{a_q}$ and the only processors with idle time are $P_{i_1}, P_{i_2}, \ldots, P_{i_\ell}$. Idle time is partitioned into disjoint regions, which we call disjoint processors. Initially we define disjoint processors $DPO_k$ for $1 \leq k \leq \ell$ as processors $P_{i_{\ell-j+1}}$ from time $\mu_{i_{\ell-j-k+2}}$ to $\mu_{i_{\ell-j-k+1}}$ for $1 \leq j \leq \ell - k$ and processor $P_{i_k}$ from time $\mu_{i_1}$ to $\beta$ (see figure 3a in the appendix). It is assumed that $\mu_{i_\ell} \leq \mu_{i_{\ell-1}} \leq \cdots \leq \mu_{i_1} \leq \beta$. The total processing capability of $DPO_k$ is $wo_k$.

$$wo_k = s_{i_k}(\beta - \mu_{i_1}) + \sum_{j=1}^{\ell-k} s_{i_{\ell-j+1}} * (\mu_{i_{\ell-j-k+1}} - \mu_{i_{\ell-j-k+2}}).$$

Claim 1: All idle time is included in the DPO's.

Claim 2: The individual idle time blocks from a DPO are nonoverlapping, i.e., they are not defined at the same time on more than one machine.

In line 5 of the algorithm, $\mu_{i_0}$ is defined in such a way that $\mu_{i_1} \leq \mu_{i_0} \leq \beta$. Using $\mu_{i_0}$ new disjoint processors are defined as follows:

$DPN_{\ell+1}$ is $DPO_\ell$ from time $\mu_{i_0}$ to $\beta$

$DPN_k$ for $1 < k \leq \ell$ is $DPO_k$ from time 0 to $\mu_{i_0}$ and $DPO_{k-1}$ from time $\mu_{i_0}$ to $\beta$.

$DPN_1$ is $DPO_1$ from time 0 to $\mu_{i_0}$.

Claim 3: $DPN_1$ terminates exactly at time $\mu_{i_0}$.

The partition of the DPN's can be translated directly into processors and their completion times as follows:

$DPN_1$ is $P_{i_j}$ from time $\mu_{i_j}$ to $\mu_{i_{j-1}}$ for $1 \leq j \leq \ell$

$DPN_k$ for $1 < k \leq \ell + 1$ is processor $P_{i_{\ell-j+1}}$ from time $\mu_{i_{\ell-j-k+2}}$ to time $\mu_{i_{\ell-j-k+1}}$ for $1 \leq j \leq \ell - k + 1$ and processor $P_{i_{k-1}}$ from time $\mu_{i_0}$ to $\beta$

(see figure 3b in the appendix).

Claim 4: All idle time is included in the DPN's.

Claim 5: The individual idle time blocks from a DPN are nonoverlapping.

$wn_i$ for $1 \leq i \leq \ell + 1$ is defined as the total processing capability of $DPN_i$. In order to simplify the proof of correctness it is convenient to partition $DPN_i$ into $DPNL_i$ and $DPNR_i$. $DPNL_i$ is $DPN_i$ from time 0 to $\mu_{i_0}$ and $DPNR_i$ is $DPN_i$ from time $\mu_{i_0}$ to $\beta$. Let $wn\ell_i$ and $wnr_i$ be the corresponding total processing capability of $DPNL_i$ and $DPNR_i$.

Claim 6: $wo_i = wn\ell_i + wnr_{i+1}$ for $1 \leq i \leq \ell$.

$\mu_{i_0}$ is selected in such a way that it is possible to schedule $\tau_{a_q}$ to complete at time $\mu_{i_0}$ and all remaining tasks can be scheduled to complete no later than time $\beta$. In order to simplify the algorithm, $\tau_{a_1}, \tau_{a_2}, \ldots, \tau_{a_{k'}}$ will be scheduled to complete at time $\beta$. $\tau_{a_1}, \tau_{a_2}, \ldots, \tau_{a_{k'}}$ and $\tau_{a_q}$ are assigned to $DPN_1, \ldots, DPN_{k'+1}$. $\mu_{i_0}$ is selected in such a way that $\sum_{j=1}^{k'+1} wn_j = t_{a_q} + \sum_{j=1}^{k'} t_{a_j}$. A problem arises in this part. The scheduling of these tasks cannot be made uniformly from left to right on these processors, but has to be performed in some random order. Initially, procedure INITIALIZE will construct a list $V_i$ for the blocks of idle time in $DPN_i$, $1 \leq i \leq k' + 1$. An entry in $V_i$ is of the form (p, i, r, f), indicating that processor p from time r to time f is part of $DPN_i$. Whenever a task is assigned to one of these blocks, the block is deleted (or part of it) and the assignment is kept in $SL_{p,d}$ or $SR_{p,d}$ depending on whether the assignment was made in the left or right hand side of the block. After all the assignments have been made, SL and SR are added to the schedule Q. Procedure TERMINATE will carry out this operation as well as updating $\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_\ell}$.

Let us consider the following simple example. Block $(p, i, r_1, f_1)$ is initially part of $V_i$. After several iterations, block $(p, i, r_2, f_2)$ could be in $V_j$, $SL_{p,i} = (k_1, r_1, r_2)$ and $SR_{p,i} = (k_2, f_2, f_1)$. This is, the original block (p, i) is now part of $DP_j$ and consists of processor $P_p$ from time $r_2$ to $f_2$. Task $\tau_{k_1}$ has been assigned to $P_p$ from time $r_1$ to $r_2$ and $\tau_{k_2}$ from time $f_2$ to $f_1$.

We now present the algorithm which we shall refer to as Uβ:OMFT.

<u>algorithm</u> Uβ:OMFT (m, n, β, Q, t, s)

//given m uniform processors denoted by $P_1, P_2, \ldots, P_m$; the algorithm constructs a β:OMFT preemptive schedule for tasks $\tau_1, \tau_2, \ldots, \tau_n$ with execution time requirements are $t_1 \leq t_2 \leq \ldots \leq t_n$. The relative speed of processor $P_i$ is $s_i$ and $s_1 \geq s_2 \geq \ldots \geq s_m$. The schedule for processor $P_j$ is represented in $Q_j$. The kth entry in $Q_j$ is of the form (i, r, f), indicating that $\tau_i$ is to be executed from time r to time f by processor $P_j$.//

//initialize the processor schedules

$\mu_j$: total time processor $P_j$ is busy.

$wo_j$: total processing capability of $DPO_j$//

1 $[\mu_i \leftarrow 0;\ Q_i \leftarrow \phi;\ wo_i \leftarrow \beta * s_i]$ for $1 \leq i \leq m$

//initialize processor and task indices//

2 $i_j \leftarrow j$ for $0 \leq j \leq m$; $\ell \leftarrow m$;

3 $a_j \leftarrow n - j + 1$ for $1 \leq j \leq n$; $q \leftarrow n$;

//schedule task $\tau_{a_q}$//

4 <u>while</u> $q \neq 0$ <u>do</u>

5 $\mu_{i_0} \leftarrow \max\limits_{0 \leq k \leq \min\{\ell-1,\, q-1\}} \{x \mid t_{a_q} + \sum\limits_{j=1}^{k} t_{a_j} = (\sum\limits_{j=1}^{k+1} wo_j) - (\beta - x)s_{i_{k+1}}\}$

6 Let k' be the maximum value of k for which x is maximum (line 5)

//Construct DPN's from DPO's//

7 $wn_{\ell+1} \leftarrow (\beta - \mu_{i_0})s_{i_\ell}$

$wn_j \leftarrow wo_j - (\beta - \mu_{i_0})(s_{i_j} - s_{i_{j-1}})$ for $2 \leq j \leq \ell$

$wn_1 \leftarrow wo_1 - (\beta - \mu_{i_0})s_{i_1}$

//initialize blocks of idle time for $DPN_1, DPN_2, \ldots, DPN_{k'+1}$//

8 INITIALIZE

//assign tasks $\tau_{a_1}, \tau_{a_2}, \ldots, \tau_{a_{k'}}$//

9 <u>for</u> $i = 1$ <u>to</u> $k'$ <u>do</u>

10 Find the minimum positive integer $p'$ be such that $wn_{j+1} > t_{a_j}$ for $1 \leq j \leq p' - 1$ and $wn_{p'+1} \leq t_{a_{p'}}$

//assign $\tau_{a_{p'}}$ to $DPN_{p'}$ and $DPN_{p'+1}$//

11 ASSIGN($a_{p'}$, $p'$, $p' + 1$)

//the unused portion of $DPN_{p'}$ and $DPN_{p'+1}$ are combined to form $DPN_{p'}$//

12 $(wn_1, wn_2, \ldots, wn_{k'+1-i}) \leftarrow (wn_1, wn_2, \ldots, wn_{p'-1}, wn_{p'} + wn_{p'+1} - t_{a_{p'}}, wn_{p'+2}, \ldots, wn_{k'+2-i})$

13 $(v_1, v_2, \ldots, v_{k'+1-i}) \leftarrow (v_1, v_2, \ldots, v_{p'-1}, v_{p'}, v_{p'+2}, \ldots, v_{k'+2-i})$

//eliminate $\tau_{a_{p'}}$//

14 $(a_1, a_2, \ldots, a_{k'-i}) \leftarrow (a_1, a_2, \ldots, a_{p'-1}, a_{p'+1}, \ldots, a_{k'-i+1})$

15 <u>endfor</u>

//assign task $\tau_{a_q}$ to $DPN_1$//

16 ASSIGN($a_q$, 1, 0)

17 TERMINATE

18 $(i_1, i_2, \ldots, i_{\ell-k'}) \leftarrow (i_{k'+1}, i_{k'+2}, \ldots, i_\ell)$

19 $(wo_1, wo_2, \ldots, wo_{\ell-k'}) \leftarrow (wn_{k'+2}, wn_{k'+3}, \ldots, wn_{\ell+1})$

20 $\ell \leftarrow \ell - k'$

21 $(a_1, a_2, \ldots, a_{q-k'-1}) \leftarrow (a_{k'+1}, a_{k'+2}, \ldots, a_{q-1})$

$q \leftarrow q - (k' + 1)$

22 <u>endwhile</u>

23 <u>end of algorithm</u> Uβ:OMFT

<u>procedure</u> INITIALIZE

//initialize blocks for $DPN_1, DPN_2, \ldots, DPN_{k'+1}$//

1 $V_1 \leftarrow (i_\ell, 1, \mu_{i_\ell}, \mu_{i_{\ell-1}})(i_{\ell-1}, 1, \mu_{i_{\ell-1}}, \mu_{i_{\ell-2}})\ldots(i_1, 1, \mu_{i_1}, \mu_{i_0})$

$V_j \leftarrow (i_\ell, j, \mu_{i_{\ell+1-j}}, \mu_{i_{\ell-j}})(i_{\ell-1}, j, \mu_{i_{\ell-j}}, \mu_{i_{\ell-j-1}})\ldots$

$(i_j, j, \mu_{i_1}, \mu_{i_0})(i_{j-1}, j, \mu_{i_0}, \beta)$ for $1 < j \leq k' + 1$

2 $SL_{k,j} \leftarrow SR_{k,j} \leftarrow \emptyset$ for $i_{j-1} \leq k \leq i_\ell$, $1 \leq j \leq k' + 1$

3 <u>end of procedure INITIALIZE</u>

<u>procedure</u> TERMINATE

//assignments in SL and SR are made final in Q//

1 $Q_{i_k} \leftarrow Q_{i_k} \| SL_{k,j} \| SR_{k,j}$ for $1 \leq j \leq \min\{k' + 1, k + 1\}$, $1 \leq k \leq \ell$

2 $(\mu_{i_{k'+1}}, \ldots, \mu_{i_\ell}) \leftarrow (\mu_{i_0}, \mu_{i_1}, \ldots, \mu_{i_{\ell-k'-1}})$

3 $\mu_{i_1} \leftarrow \mu_{i_2} \leftarrow \ldots \leftarrow \mu_{i_{k'}} \leftarrow \beta$

4 <u>end of procedure</u> TERMINATE

<u>procedure</u> ASSIGN (j, p, r)

//assign $\tau_j$ to $DPN_p$ from time 0 to b and to $DPN_r$ from time b to β. In case $r = 0$, b is set to β (i.e., $V_0 = \emptyset$).//

1 Let b be such that the idle time on $DPN_p$ from time 0 to b and in $DPN_r$ from time b to β is $t_j$. In case $r = 0$, b is set to β.

//B is defined as $DPN_r$ from time 0 to b//

```
B ← ∅;
while V_r ≠ ∅ and b > initial time for the first tuple in V_r do
     (pn, dp, initial, final) ⇐ V_r  //the leftmost tuple in V_r is
     deleted and assigned to (.,.,.,.)//
     B ← B||(pn, dp, initial, min{final, b})
     if b = min{final, b} then [if b < final then
                                 [SR_pn,dp ← (j, b, final)||SR_pn,dp]
                           exit while loop]
endwhile
//assign τ_j to the remaining blocks of DPN_r//
while V_r ≠ ∅ do
     (pn, dp, initial, final) ⇐ V_r
     SL_pn,dp ← SL_pn,dp||(j, initial, final)
endwhile
//assign τ_j to DPN_p from time 0 to b//
while V_p ≠ ∅ and b > initial time for the first tuple in V_p do
     (pn, dp, initial, final) ⇐ V_p
     SL_pn,dp ← SL_pn,dp||(j, initial, min{final, b})
     if b = min{final, b} then [if b < final then
                                 [B ← B||(pn, dp, b, final)]
                           exit while loop]
endwhile
V_p ← B||V_p
end of procedure ASSIGN
```

Example 1: Given a deadline $\beta = 35$, 8 tasks with execution time $t_1 = 3$, $t_2 = 11$, $t_3 = 13$, $t_4 = 25$, $t_5 = 26$, $t_6 = 29$, $t_7 = 31$, $t_8 = 72$, and 3 uniform machines with relative speeds $s_1 = 3$, $s_2 = 2$ and $s_3 = 1$. Algorithm Uβ:OMFT constructs a β:OMFT preemptive schedule as follows:

| $(a_1,a_2,\ldots,a_q)$ | (q) | $(i_1,i_2,i_\ell)$ | ($\ell$) | $(\mu_1,\mu_2,\mu_3)$ | (schedule*) | step |
|---|---|---|---|---|---|---|
| (8,7,6,5,4,3, 2,1) | (8) | (1,2,3) | (3) | (0,0,0) | $Q_1 = Q_2 = Q_3 = \phi$ | Initial Conditions |
| | | | k' = 0 | | | |
| (8,7,6,5,4,3, 2) | (7) | (1,2,3) | (3) | (1,0,0) | $Q_1$ = [(1,0,1)]<br>$Q_2 = Q_3 = \phi$ | end of 1st iteration |
| | | | k' = 0 | | | |
| (8,7,6,5,4,3) | (6) | (1,2,3) | (3) | (4,1,0) | $Q_1$ = [(1,0,1), (2,1,4)]<br>$Q_2$ = [(2,0,1)]<br>$Q_3 = \phi$ | end of 2nd iteration |
| | | | k' = 0 | | | |
| (8,7,6,5,4) | (5) | (1,2,3) | (3) | (6,4,1) | $Q_1$ = [(1,0,1) (2,1,4),(3,4,6)]<br>$Q_2$ = [(2,0,1), (3,1,4)]<br>$Q_3$ = [(3,0,1)] | end of 3rd iteration |

*The tuple (i, s, f) in $Q_j$ indicates that processor $P_j$ will execute task $\tau_i$ from time s to time f. The tuples with s = f have been eliminated.

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | k' = 0 | | | |
| (8,7,6,5) | (4) | (1,2,3) | (3) | (12,6,4) | $Q_1$ = [(1,0,1), (2,1,4), (3,4,6), (4,6,12)]<br>$Q_2$ = [(2,0,1), (3,1,4), (4,4,6)]<br>$Q_3$ = [(3,0,1), (4,1,4)] | end of 4th iteration |
| | | | k' = 1 | | | |
| (7,6) | (2) | (2,3) | (2) | (35,16.5,12) | $Q_1$ = [(1,0,1), (2,1,4), (3,4,6) (4,6,12), (5,12,16.5), (8,16.5,35)]<br>$Q_2$ = [(2,0,1), (3,1,4), (4,4,6), (5,6,12), (8, 12, 16.5)]<br>$Q_3$ = [(3,0,1), (4,1,4), (8,4,5.5), (5,5.5,6), (8,6,12)]<br>(see figure 1a) | end of 5th iteration |
| | | | k' = 1 | | | |
| (∅) | (0) | (3) | (1) | (35,35,35) | $Q_1$ = [(1,0,1), (2,1,4), (3,4,6), (4,6,12), (5,12,16.5), (8,16.5,35)]<br>$Q_2$ = [(2,0,1), (3,1,4), (4,4,6), (5,6,12), (8,12,16.5), (7,16.5,24.5), (6,24.5,35)]<br>$Q_3$ = [(3,0,1), (4,1,4) (8,4,5.5), (5,5.5,6), (8,6,12), (7,12,16.5), (6,16.5,24.5), (7,24.5,35)]<br>(see figure 1b) | end of 6th iteration |

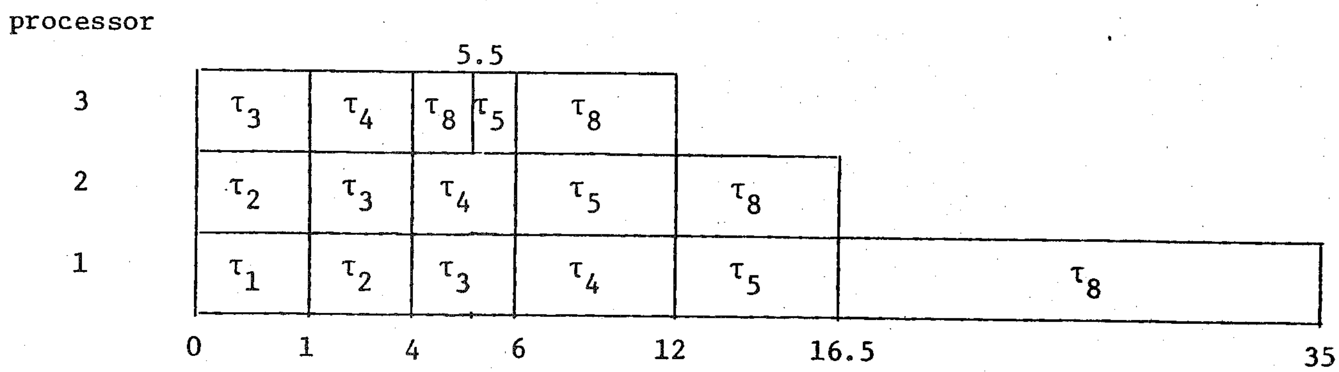


Figure 1a: Schedule for $\tau_1$, $\tau_2$, $\tau_3$, $\tau_4$, $\tau_5$ and $\tau_8$

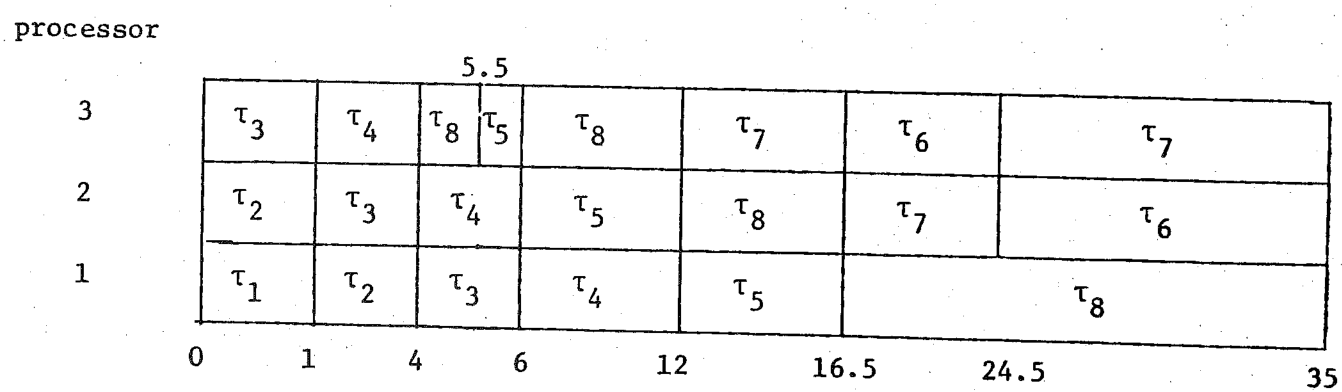


Figure 1b: Final Schedule for example 1

Before proving the correctness of the algorithm we establish some useful properties. At some point during the exeuction of the algorithm $U\beta$:OMFT, all or some of the following properties will hold true.

h1) $Q$ is a schedule for tasks $\tau_j$, $j \epsilon N-A$

h2) $\mu_{i_\ell} \le \mu_{i_{\ell-1}} \le \dots \le \mu_{i_1} \le \beta$

$\mu_{i_j} = \beta$ for $j \epsilon M-I$

h3) $\sum_{j=1}^{k} t_{a_j} \le \sum_{j=1}^{k} wo_j$ for $1 \le k \le \min\{\ell, q\}$

h4) $\sum_{j=1}^{m} \mu_j s_j = \sum_{j \varepsilon N-A} t_j$ (note that $\sum_{j \varepsilon \phi} t_j = 0$)

h5) $t_{a_j} \geq t_{a_{j+1}}$ for $1 \leq j < q$

h6) $a_j - 1 = a_{j+1}$ for $1 \leq j < q$

$a_q = v + 1$, where $v$ is the number of times loop 4-22 (Uβ:OMFT) has been executed ($a_0 = r + 1$, where $r$ is the total number of times loop 4-22 was executed. Assume $t_{n+1} \geq t_n$).

h7) $wo_1 - (\beta - \mu_{i_1})s_{i_1} \leq t_{a_q}$

h8) $f_{a_q - j} = \mu_{i_j}$ for $1 \leq j \leq \ell$ (note that $f_j = 0$ for $j \leq 0$)

h9) $\ell > 0$ and $i_j = m - \ell + j$ for $1 \leq j \leq \ell$

In lemma 1, we show that h1-h9 will hold true at the beginning of each iteration (Uβ:OMFT just before line 5). This will be of use in theorem 1, where we show that algorithm Uβ:OMFT constructs β:OMFT preemptive schedules. The proof for theorem 1 also uses lemmas 2-4. Lemmas 2 and 3 study some properties of general preemptive schedules. Theorem 2 shows that the time complexity for algorithm Uβ:OMFT is O(nm). Finally, in theorem 3 it is shown that algorithm Uβ:OMFT constructs preemptive schedules with no more than O(nm) preemptions.

Lemma 1: At the beginning of each iteration (Uβ:OMFT just before line 5), h1-h9 will hold true.

Proof: It is simple to verify that the lemma is true at the beginning of the first iteration. In order to complete the proof of the lemma it is only required to show that if h1-h9 hold true and $q \neq 0$ just before line 5 then after the execution of lines 5-22, h1-h9 will hold true.

The proof is in five separate parts. We now prove each part separately. Let $W = \{a_1, a_2, \ldots, a_{k'}, a_q\}$ after line 8 has been executed. Note that the set $W$ contains task indices, not the symbols $a_i$.

Part 1: After the execution of line 8; b1-b2, b4-b6 and b8-b18 hold true.

b1) $Q$ is a schedule for $\tau_j$, $j \varepsilon N - (\{a_{k'+1}, \ldots, a_{q-1}\} \cup W)$

b2) $\mu_{i_\ell} \leq \mu_{i_{\ell-1}} \leq \cdots \leq \mu_{i_1} \leq \mu_{i_0} \leq \beta$

$\mu_j = \beta$ for $j \varepsilon M - \{i_1, i_2, \ldots, i_\ell\}$

b4) $\sum_{j=1}^{m} \mu_j s_j = \sum_{j \varepsilon N - (\{a_{k'+1}, \ldots, a_{q-1}\} \cup W)} t_j$

b5) $t_{a_j} \geq t_{a_{j+1}}$ for $k' + 1 \leq j < q$

b6) $a_j - 1 = a_{j+1}$ for $k' + 1 \leq j < q - 1$

$a_{q-1} = v$ ($v$ as defined in h6)

b8) $f_{a_q - j} = \mu_j$ for $1 \leq j \leq \ell$

b9) $\ell > 0$ and $i_j = m - \ell + j$ for $1 \leq j \leq \ell$

b10) $\sum_{j=k'+1}^{k} wn_{j+1} \geq \sum_{j=k'+1}^{k} t_{a_j}$ for $k' + 1 \leq k \leq \min\{q - 1, \ell\}$

b11) SL and SR are empty schedules

b12) $t_{a_1} \geq t_{a_2} \geq \cdots \geq t_{a_{k'}} \geq t_{a_q}$

b13) $t_{a_q} + \sum_{j=1}^{k'} t_{a_j} = \sum_{j=1}^{k'+1} wn_j$

b14) $\sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k} (wn\ell_j + wnr_{j+1})$ for $1 \leq k \leq k'$

b15) $DPN_1$ terminates exactly at time $\mu_{i_0}$.

b16) $t_{a_q} + \sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k+1} wn_j$ for $0 \leq k \leq k'$

b17) All idle time blocks in $DPN_k$ $(1 \leq k \leq k' + 1)$ are nonoverlapping.

b18) let $z$, $p$ and $r$ be such that

i) $0 \leq z \leq \beta$ and $1 < p < r \leq k' + 2$

or ii) $0 \leq z \leq \mu_{i_0}$ and $1 = p < r \leq k' + 2$.

At time $z$, if $DPN_r$ has a block of idle time then so does $DPN_p$. If $DPN_p$ has a block of idle time at time $z$, then the relative speed of the processor over which it is defined is not slower than the one used by $DPN_r$ (if any).

<u>Proof of Part 1</u>: The proof is given in 1.1-1.5.

1.1) The proof for b1, b4-b6, b8-b9 and b12 follow from h1, h4-h6, h8-h9. Note that the algorithm does not modify the variables used.

1.2) In order to prove b2, it is only required to show that $\mu_{i_0} \geq \mu_{i_1}$ and $\mu_{i_0} \leq \beta$ after line 5.

i) <u>$\mu_{i_0} \geq \mu_{i_1}$</u>:

From line 5, together with $q \neq 0$ and $\ell > 0$ (h9) we have that

$$\mu_{i_0} \geq x = (t_{a_q} - wo_1 + \beta s_{i_1})/s_{i_1}.$$

Substituting $t_{a_q} \geq wo_1 - (\beta - \mu_{i_1})s_{i_1}$ (h7)

we obtain $\mu_{i_0} \geq \mu_{i_1}$.

ii) <u>$\mu_{i_0} \leq \beta$</u>:

There are two cases depending on the value of $\ell$.

<u>case 1: $\ell = 1$</u>

Substituting $\mu_j = \beta$ for $j\varepsilon M - \{i_1\}$ (h2) in $\sum_{j=1}^{m} \mu_j s_j = \sum_{j\varepsilon N-\{a_1,a_2,\ldots,a_q\}} t_j$ (h4) we obtain $\mu_{i_1} s_{i_1} + \sum_{j\varepsilon M-\{i_1\}} \beta s_j = \sum_{j\varepsilon N-\{a_1,\ldots,a_q\}} t_j$. Substituting the initial condition $\beta \geq (1)^{T/S_m}$, in the above equation we obtain

$$\mu_{i_1} s_{i_1} + (1)^T - \beta s_{i_1} \leq \sum_{j\varepsilon N-\{a_1,\ldots,a_q\}} t_j .$$

This can be written as $\sum_{j\varepsilon\{a_1,\ldots,a_q\}} t_j \leq s_{i_1}(\beta - \mu_{i_1})$. As $\ell = 1$ then $wo_1$ is $s_{i_1}(\beta - \mu_{i_1})$. Clearly $t_{a_q} \leq \sum_{j\varepsilon\{a_1,\ldots,a_q\}} t_j$. Substituting in the above inequality we obtain

$$t_{a_q} \leq wo_1 \tag{1}$$

As $q \neq 0$ (line 4) and $\ell = 1$, then from line 5 we have that $t_{a_q} = wo_1 - (\beta - \mu_{i_0})s_{i_1}$ or $\mu_{i_0} = (t_{a_q} - wo_1)/s_{i_1} + \beta$. Substituting (1), we get $\mu_{i_0} \leq \beta$.

<u>case 2: $\ell > 1$</u>

From lines 5 and 6 we have,

$$\mu_{i_0} = (t_{a_q} + \sum_{j=1}^{k'} t_{a_j} - \sum_{j=1}^{k'+1} wo_j)/s_{i_{k'+1}} + \beta$$

$$\text{for some } k' \text{ in } [0; \min\{\ell - 1, q - 1\}]. \tag{2}$$

From (h3) we have that

$$\sum_{j=1}^{k'+1} t_{a_j} \leq \sum_{j=1}^{k'+1} wo_j \tag{3}$$

Substituting $t_{a_q} \leq t_{a_{k'+1}}$ (h5) in (3) and then in (2) we obtain, $\mu_{i_0} \leq \beta$.

Hence, b2 holds true after line 8.

1.3) Before proving b10, b13 and b16, we prove that after the execution of line 7

$$\sum_{j=1}^{k+1} wn_j = \left(\sum_{j=1}^{k+1} wo_j\right) - (\beta - \mu_{i_0})s_{i_{k+1}} \quad \text{for} \quad 0 \leq k \leq \min\{\ell - 1, q - 1\}. \tag{4}$$

For $k = 0$, we have that $wn_1 = wo_1 - (\beta - \mu_{i_0})s_{i_1}$, which follows directly from line 7. Suppose now,

$$\sum_{j=1}^{k} wn_j = \left(\sum_{j=1}^{k} wo_j\right) - (\beta - \mu_{i_0})s_{i_k} \tag{5}$$

for some $k \geq 0$. We now prove that

$$\sum_{j=1}^{k+1} wn_j = \left(\sum_{j=1}^{k+1} wo_j\right) - (\beta - \mu_{i_0})s_{i_{k+1}} \quad \text{for} \quad 0 < k \leq \min\{\ell - 1, q - 1\}.$$

Adding $wn_{k+1}$ to (5) we obtain

$$\left(\sum_{j=1}^{k} wn_j\right) + wn_{k+1} = \left(\sum_{j=1}^{k} wo_j\right) - (\beta - \mu_{i_0})s_{i_k} + wn_{k+1}$$

replacing $wn_{k+1}$ for the operation in line 7 we get

$$\sum_{j=1}^{k+1} wn_j = \left(\sum_{j=1}^{k} wo_j\right) - (\beta - \mu_{i_0})s_{i_k} + wo_{k+1} - (\beta - \mu_{i_0})(s_{i_{k+1}} - s_{i_k})$$

$$= \left(\sum_{j=1}^{k+1} wo_j\right) - (\beta - \mu_{i_0})s_{i_{k+1}}$$

Hence,

$$\sum_{j=1}^{k+1} wn_j = \left(\sum_{j=1}^{k+1} wo_j\right) - (\beta - \mu_{i_0})s_{i_{k+1}} \quad \text{for} \quad 0 \leq k \leq \min\{\ell - 1, q - 1\}$$

Let us now prove, b10, b13 and b16. After the execution of line 6, we have from line 5 that,

$$t_{a_q} + \sum_{j=1}^{k} t_{a_j} \le \left(\sum_{j=1}^{k+1} wo_j\right) - (\beta - \mu_{i_0}) s_{i_{k+1}} \quad \text{for} \quad 0 \le k \le k' \tag{6}$$

$$t_{a_q} + \sum_{j=1}^{k'} t_{a_j} = \left(\sum_{j=1}^{k'+1} wo_j\right) - (\beta - \mu_{i_0}) s_{i_{k'+1}} \tag{7}$$

and, $t_{a_q} + \sum_{j=1}^{k} t_{a_j} < \left(\sum_{j=1}^{k+1} wo_j\right) - (\beta - \mu_{i_0}) s_{i_{k+1}}$

$$\text{for} \quad k' + 1 \le k \le \min\{\ell - 1, q - 1\}. \tag{8}$$

After line 7 we substitute (4) in (6), (7) and (8)

$$t_{a_q} + \sum_{j=1}^{k} t_{a_j} \le \sum_{j=1}^{k+1} wn_j \quad \text{for} \quad 0 \le k \le k'$$

$$t_{a_q} + \sum_{j=1}^{k'} t_{a_j} = \sum_{j=1}^{k'+1} wn_j$$

$$t_{a_q} + \sum_{j=1}^{k} t_{a_j} < \sum_{j=1}^{k+1} wn_j \quad \text{for} \quad k' + 1 \le k \le \min\{q - 1, \ell - 1\}.$$

Subtracting the second equation from the third inequality,

$$\sum_{j=k'+1}^{k} t_{a_j} < \sum_{j=k'+2}^{k+1} wn_j \quad \text{for} \quad k' + 1 \le k \le \min\{q - 1, \ell - 1\}.$$

Using the initial condition $(1)\ T \le \beta S_m$ and h4 we get

$$\sum_{j=k'+1}^{k} t_{a_j} \le \sum_{j=k'+2}^{k+1} wn_j \quad \text{for} \quad k' + 1 \le k \le \min\{q - 1, \ell\}$$

Hence, b10, b13 and b16 hold true after line 8.

1.4) The proof for b14 follows from h3 together with claim 6. In line 1 (INITIALIZE), all idle time blocks in $V_1$ have finish time $\le \mu_{i_0}$.

Therefore, b15 holds true after line 8. SL and SR are initialized as empty schedules (line 2 of procedure INITIALIZE). So, b11 holds true after line 8. From claim 5 (or INITIALIZE line 1), it follows that b17 holds true after line 8.

1.5) We now prove b18. From time $\mu_{i_0}$ to $\beta$, $DPN_k$ is defined on $P_{i_{k-1}}$ for $1 < k \leq \ell + 1$. Using h9 together with the initial conditions $s_j \geq s_{j+1}$ for $1 \leq j < m$, it is simple to show that b18 holds true from time $\mu_{i_0}$ to $\beta$. Let us now consider any point in time from $\mu_{i_j}$ to $\mu_{i_{j-1}}$ $(1 \leq j \leq \ell)$. $DPN_1$ is defined on $P_{i_j}$ and $DPN_k$ $(1 < k \leq \ell + 1)$ is defined over $P_{i_{j+k-1}}$. This together with the initial condition $s_j \geq s_{j+1}$ for $1 \leq j < m$ and h9 imply that b18 holds true from time 0 to $\mu_{i_0}$. Hence, b18 holds true after line 8.

This completes the proof for part 1. □

Let us consider loop 9-25. Let i be the value of i in line 9.

<u>Part 2</u>: c1 and c11-c18 hold true after i gets the value of one or i is increased by one (before the test in line 9 is performed).

c1) Same as b1-b2, b4-b6 and b8-b10

c11) SL and SR are the schedules for $\tau_k$, $k \in W - \{a_1, \ldots, a_{k'-i+1}, a_q\}$

c12) $t_{a_1} \geq \ldots \geq t_{a_{k'-i+1}} \geq t_{a_q}$

c13) $t_{a_q} + \sum_{j=1}^{k'-i+1} t_{a_j} = \sum_{j=1}^{k'-i+2} wn_j$

c14) $\sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k} (wn\ell_j + wnr_{j+1})$ for $1 \leq k \leq k' - i + 1$

c15) $DPN_1$ terminates exactly at time $\mu_{i_0}$.

c16) $t_{a_q} + \sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k+1} wn_j$ for $0 \leq k \leq k' - i + 1$

c17) All idle time blocks on $DPN_k$ $(1 \leq k \leq k' - i + 2)$ are nonoverlapping.

c18) Let $z$, $p$ and $r$ be such that

i) $0 \leq z \leq \beta$ and $1 < p < r \leq k' - i + 2$

ii) $0 \leq z \leq \beta$ and $1 < p < r = k' + 2$

iii) $0 \leq z \leq \mu_{i_0}$ and $1 = p < r \leq k' - i + 2$ or

iv) $0 \leq z \leq \mu_{i_0}$ and $1 = p < r = k' + 2$

At time $z$, if $DPN_r$ has a block of idle time then so does $DPN_p$. If $DPN_p$ is defined at time $z$, it is defined over a processor whose speed is not slower than the one used by $DPN_r$ (if any).

Proof of Part 2:

For $i = 1$, c1 and c11-c18 follow b1-b2, b4-b6 and b8-b18. In order to complete the proof it is only required to show that if c1 and c11-c18 hold true the $i^{th}$ time ($i \neq k' + 1$) line 9 is executed, then after lines 9-15 are executed and $i$ is increased, c1 and c11-c18 will hold true. Let c'1 and c'11-c'18 denote c1 and c11-c18 before the loop is executed.

In line 10, a value for $p'$ in the range $[1; k' - i + 1]$ will always be found, as $t_{a_q} \leq wn_1$ (c'16) and $t_{a_q} + \sum_{j=1}^{k'-i+1} t_{a_j} = \sum_{j=1}^{k'-i+2} wn_j$ (c'13).

Consider now the call to procedure ASSIGN (line 11). First let us show that a value for $b$ will always exist in line 1.

case 1: $p' > 1$

Let $b_0$, $b_1$, $b_2$ and $b_3$ be such that $b_0 = 0 \leq b_1 < b_2 \leq \beta = b_3$. Let $c_i$ be the processing capability of $DPN_{p'}$ from time 0 to $b_i$ and on $DPN_{p'+1}$ from time $b_i$ to $\beta$. Clearly $c_1 \leq c_2$ (this follows from c'18). From line 10 (Uβ:OMFT) we have that $t_{a_{p'-1}} < wn_{p'}$ (or $t_{a_{p'-1}} < c_3$) and $t_{a_{p'}} \geq wn_{p'+1}$ (or $t_{a_{p'}} \geq c_0$). As

$t_{a_{p'}} \leq t_{a_{p'-1}}$ (c'12), it must be that $c_3 > t_{a_{p'}}$. Hence, a value for b will always be found in line 1.

case 2: $p' = 1$

Let $b_0$, $b_1$, $b_2$ and $b_3$ be such that $b_0 = 0 \leq b_1 < b_2 \leq \mu_{i_0} = b_3$. Let $c_i$ be the processing capability of $DPN_1$ from time 0 to $b_i$ and on $DPN_2$ from time $b_i$ to β. Clearly $c_1 < c_2$ (this follows from c'18). Now, $t_{a_1} \geq wn_2$ (line 10, Uβ:OMFT) and $t_{a_1} \leq wn\ell_1 + wnr_2$(c'14). So, $c_3 \geq t_{a_1}$ and $c_0 \leq t_{a_1}$. Hence, there is always a point b such that $0 \leq b \leq \mu_{i_0}$.

The remaining part of procedure ASSIGN will schedule $\tau_{a_{p'}}$ from time 0 to b on $DPN_{p'}$ and from time b to β on $DPN_{p'+1}$. $DPN_{p'}$ is then redefined as $DPN_{p'+1}$ from time 0 to b and $DPN_{p'}$ from time b to β. Lines 12-13 in Uβ:OMFT renames $DPN_{p'+2}, \ldots, DPN_{k'+2-i}$ as $DPN_{p'+1}, \ldots, DPN_{k'+1-i}$. $\tau_{a_{p'}}$ is eliminated in line 14.

As $p' \leq k' - i + 1$, none of the variables in c'1 are modified. So, c1 follows from c'1. The proofs for c11-c13, c15 and c17-c18 are simple and will be omitted. In 2.1 and 2.2 we prove c16 and c14.

2.1) It is now required to show that c16 will hold true after i is increased in line 9. Initially

$$t_{a_q} + \sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k+1} wn_j \quad \text{for} \quad 0 \leq k \leq k' - i + 1 \qquad (c'16)$$

This can be broken into

$$t_{a_q} + \sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k+1} wn_j \quad \text{for} \quad 0 \leq k < p' - 1 \qquad (9)$$

and $$t_{a_q} + \sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k+1} wn_j \quad \text{for} \quad p' \leq k \leq k' - i + 1 \tag{10}$$

The body of the loop modifies $a_i$ and $wn_i$ (let $a'$ and $wn'$ be the new values for $a$ and $wn$) as follows:

$$(a'_1, a'_2, \ldots, a'_{k'-i}, a'_q) \leftarrow (a_1, \ldots, a_{p'-1}, a_{p'+1}, \ldots, a_{k'-i+1}, a_q)$$

$$(wn'_1, wn'_2, \ldots, wn'_{k'-i+1}) \leftarrow (wn_1, \ldots, wn_{p'-1}, wn_{p'} + wn_{p'+1} - t_{a_{p'}}, wn_{p'+2}, \ldots, wn_{k'-i+2})$$

Substituting in (9) and (10) we obtain a) and b).

a) $$t_{a'_q} + \sum_{j=1}^{k} t_{a'_j} \leq \sum_{j=1}^{k+1} wn'_j \quad \text{for} \quad 0 \leq k < p' - 1 \tag{11}$$

and b) $$t_{a'_q} + \left(\sum_{j=1}^{p'-1} t_{a'_j}\right) + t_{a_{p'}} + \sum_{j=p'}^{k} t_{a'_j} \leq \left(\sum_{j=1}^{p'-1} wn'_j\right) + wn_{p'} + wn_{p'+1} + \sum_{j=p'+1}^{k+1} wn'_j \quad \text{for} \quad p' - 1 \leq k < k' - i + 1$$

$$t_{a'_q} + \sum_{j=1}^{k} t_{a'_j} \leq \sum_{j=1}^{k+1} wn'_j \quad \text{for} \quad p' - 1 \leq k \leq k' - i \tag{12}$$

After $i$ is increased, c16 follows from (11) and (12).

2.2) Let us now prove c14. Initially,

$$\sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k} (wn\ell_j + wnr_{j+1}) \quad \text{for} \quad 1 \leq k \leq k' - i + 1 \tag{c'14}$$

<u>case 1</u>: $b \leq \mu_{i_0}$ in line 1 (ASSIGN)

The above inequalities can be broken into:

$$\sum_{j=1}^{k} t_{a_j} \leq \sum_{j=1}^{k} (wn\ell_j + wnr_{j+1}) \quad \text{for} \quad 1 \leq k \leq p' - 1 \tag{13}$$

and $\sum_{j=1}^{k} t_{a_j} \le \sum_{j=1}^{k} (wn\ell_j + wnr_{j+1})$ for $p' + 1 \le k \le k' - i + 1$ (14)

Since $b \le \mu_{i_0}$, the algorithm modifies* the values for wnr, wnℓ and a (a', wnℓ' and wnr' are the new values for a, wnℓ and wnr) as follows:

$$(wn\ell'_1, \ldots, wn\ell'_{k'-i}) \leftarrow (wn\ell_1, \ldots, wn\ell_{p'-1}, wn\ell_{p'} + wn\ell_{p'+1} + wnr_{p'+1} - t_{a_{p'}}, wn\ell_{p'+2}, \ldots, wn\ell_{k'-i+1})$$

$$(wnr'_1, \ldots, wnr'_{k'-i+1}) \leftarrow (wnr_1, \ldots, wnr_{p'}, wnr_{p'+2}, \ldots, wnr_{k'-i+2})$$

$$(a'_1, \ldots, a'_{k'-i}, a'_q) \leftarrow (a_1, \ldots, a_{p'-1}, a_{p'+1}, \ldots, a_{k'-i+1}, a_q)$$

Substituting in (13) and (14) we obtain a) and b).

a) $\sum_{j=1}^{k} t_{a'_j} \le \sum_{j=1}^{k} (wn\ell'_j + wnr'_{j+1})$ for $1 \le k \le p' - 1$ (15)

and b) $\sum_{j=1}^{k} t_{a'_j} + t_{a_{p'}} \le \sum_{j=1}^{p'-1} (wn\ell'_j + wnr'_{j+1}) + \sum_{j=p'+1}^{k} (wn\ell'_j + wnr'_{j+1}) + wn\ell_{p'} + wn\ell_{p'+1} + wnr_{p'+1} + wnr_{p'+2}$ for $p' \le k \le k' - i$

$\sum_{j=1}^{k} t_{a'_j} \le \sum_{j=1}^{k} (wn\ell'_j + wnr'_{j+1})$ for $p' \le k \le k' - i$. (16)

After increasing the value for i, c14 follows from (15) and (16).

case 2: $b > \mu_{i_0}$ in line 1 (ASSIGN).

The proof is similar to the one for case 1 and will be omitted.

This completes the proof of part 2. □

*Note that the algorithm only modifies wn, but this changes wnℓ and wnr.

Part 3: Just before the execution of line 16, d1, d11, d13, d15, d17 and d18 will hold true.

d1) Same as b1-b2, b4-b6 and b8-b10.

d11) SL and SR is a schedule for $\tau_k$, $k \epsilon W - \{a_q\}$.

d13) $t_{a_q} = wn_1$.

d15) $DPN_1$ terminates exactly at time $\mu_{i_0}$.

d17) All idle time blocks on $DPN_1$ are nonoverlapping.

d18) Let z, p and r be such that

$$0 \leq z \leq \mu_{i_0}, \quad p = 1 \quad \text{and} \quad r = k' + 2$$

At time z, if $DPN_r$ has a block of idle time then so does $DPN_p$. If $DPN_p$ is defined at time z, it is defined over a processor whose speed is not slower than the one used by $DPN_r$ (if any).

Proof of Part 3:

d1, d11, d13, d15, d17 and d18 follows from c1, c11, c13, c15, c17 and c18 together with the observation that the last value for i in line 9 is k' + 1. □

Part 4: Just after the execution of line 16, f1, f11, f13 and f16 will hold true.

f1) Same as b1-b2, b4-b6 and b8-b10.

f11) SL and SR is a schedule for $\tau_k$, $k \epsilon W$.

f13) $f_{a_q} = \mu_{i_0}$.

f16) $t_{a_{q-1}} \geq wn\ell_{k'+2}$.

Proof of Part 4:

The variables in d1 are not modified, it then follows that f1 is true after line 16. Procedure ASSIGN schedules task $\tau_{a_q}$. This, together with d11, imply that f11 holds true after line 16. Since $t_{a_q} = wn_1$ (d13), $DPN_1$ terminates at time $\mu_{i_0}$ (d15) and in line 16 task $\tau_{a_q}$ is assigned to $DPN_1$, it then follows that the finish time for task $\tau_{a_q}$ is $\mu_{i_0}$ (f13). From d18 and d13 we have that $t_{a_q} \geq wn\ell_{k'+2}$. As $t_{a_{q-1}} \geq t_{a_q}$, it then follows that $t_{a_{q-1}} \geq wn\ell_{k'+2}$ (f16).

This completes the proof for part 4. □

Part 5: Just after the execution of line 17, g1-g2, g4-g6, g8-g10 and g16 hold true.

g1) Q is a schedule for tasks $\tau_k$, $k \in N - \{a_{k'+1}, \ldots, a_{q-1}\}$

g2) $\beta \geq \mu_{i_{k'+1}} \geq \mu_{i_{k'+2}} \geq \ldots \geq \mu_{i_\ell}$

$\mu_j = \beta$ for $j \in M - \{i_{k'+1}, \ldots, i_\ell\}$

g4) $\sum_{j=1}^{m} \mu_j s_j = \sum_{j \in N - \{a_{k'+1}, \ldots, a_{q-1}\}} t_j$

g5) $t_{a_{k'+1}} \geq \ldots \geq t_{a_{q-1}}$

g6) $a_i + 1 = a_{i+1}$ for $k' + 1 \leq i < q - 1$

$a_{q-1} = v$ (v as defined in h6)

g8) $f_{a_q + k' + 1 - j} = \mu_{i_j}$ for $k' + 1 \leq j \leq \ell$

g9) $\ell > 0$ and $i_j = m - \ell + j$ for $1 \leq j \leq \ell$

g10) $\sum_{j=k'+1}^{k} wn_{j+1} \geq \sum_{j=k'+1}^{k} t_{a_j}$ for $k' + 1 \leq k \leq \min\{q - 1, \ell\}$

g16) $t_{a_{q-1}} \geq wn\ell_{k'+2}$

g5-g6, g9-g10 and g16 follow directly from f1 and f16. Using f1 and f11, together with the effect of procedure TERMINATE (the assignments in SL and SR are made final in Q), it then follows that g1 holds true after line 17. Before line 8, $DPN_1, \ldots, DPN_{k'+1}$ had the same processing capability as the processing requirements of $\tau_{a_1}, \ldots, \tau_{a_{k'}}$ and $\tau_{a_q}$. As all of these tasks have been scheduled on $DPN_1, \ldots, DPN_{k'+1}$ and $\mu_{i_1}, \ldots, \mu_{i_\ell}$ have been set to their new values, it follows that g2 and g4 hold true after line 17. g8 follows from f8 (see f1) together with the renaming of $\mu_{i_1}, \ldots, \mu_{i_\ell}$. This completes the proof of part 5. □

Part 6: h1-h9 hold true after line 21 is executed.

The proof for this part is simple and will be omitted. The proof for h3 follows from g10 and the one for h7 from g16.

Hence, h1-h9 hold true each time line 4 is executed. This completes the proof of the lemma. □

Theorem 1: For every system of $m \geq 1$ uniform processors, $n \geq m$ independent tasks and a deadline $\beta$,

$$\beta \geq \max\{ \max_{1 \leq j \leq m} \{ {}_{(n-j+1)}T/S_j \}, {}_{(1)}T/S_m \},$$

algorithm Uβ:OMFT constructs β:OMFT preemptive schedules.

Proof: First of all we prove some properties of the schedule produced by the algorithm. Then we show that such a schedule is a β:OMFT preemptive schedule.

Let S be the schedule constructed by algorithm Uβ:OMFT (note that the algorithm terminates after at most n iterations). The last time line 4 is executed, it must have been that q was zero. At this point h1-h9 (lemma 1) hold true. So, it must be that S is a feasible schedule (h1) and $ft(S) \leq \beta$ (h2). In order to prove the theorem we obtain some inequalities that relate finishing times to execution times.

Let r be the number of times loop 4-22 was executed. Let $\ell_k$ represent the value of $\ell$ at the end of the $k^{th}$ iteration. At the end of the $k^{th}$ iteration $(1 \leq k \leq r)$, we have that h1-h9 hold true (lemma 1). Clearly,

$$a_q = k + 1 \qquad \text{(h6)} \tag{17}$$

and $$f_{a_q - j} = \mu_{i_j} \quad \text{for} \quad 1 \leq j \leq \ell_k \qquad \text{(h8)} \tag{18}$$

Note that $f_j = 0$ for $j \leq 0$. Substituting (17) in (18), multiplying each side by $s_{i_j}$ and adding all equations we obtain (19).

$$\sum_{j=1}^{\ell_k} f_{k+1-j}\, s_{i_j} = \sum_{j=1}^{\ell_k} \mu_{i_j} s_{i_j} \tag{19}$$

Now, $\mu_j = \beta$ for $j \varepsilon M - \{i_1, i_2, \ldots, i_{\ell_k}\}$ (h2). Multiplying both sides by $s_j$ and adding all terms, we obtain (20)

$$\sum_{j \varepsilon M - \{i_1, \ldots, i_{\ell_k}\}} \mu_j s_j = \sum_{j \varepsilon M - \{i_1, \ldots, i_{\ell_k}\}} s_j \beta \tag{20}$$

Adding (19) and (20)

$$\sum_{j \varepsilon M - \{i_1, \ldots, i_{\ell_k}\}} s_j \beta + \sum_{j=1}^{\ell_k} f_{k+1-j}\, s_{i_j} = \sum_{j=1}^{m} \mu_j s_j \tag{21}$$

Since $i_j = m - \ell_k + j$ for $1 \leq j \leq \ell_k$ (h9), it follows that $M - \{i_1, \ldots, i_{\ell_k}\} = \{1, 2, \ldots, m - \ell_k\}$. Substituting in (21)

$$(S_{m-\ell_k})\beta + \sum_{j=1}^{\ell_k} f_{k+1-j}\ s_{m-\ell_k+j} = \sum_{j=1}^{m} \mu_j s_j$$

where $S_j = \sum_{i=1}^{j} s_i$. Substituting h4 we obtain (22)

$$(S_{m-\ell_k})\beta + \sum_{j=1}^{\ell_k} f_{k+1-j}\ s_{m-\ell_k+j} = \sum_{j\varepsilon N-\{a_1,\ldots,a_q\}} t_j \qquad (22)$$

As $a_q = k + 1$ (17), it then follows from h6 that $a_1 = k + q$. q was initially n, but every iteration (loop 4-22, Uβ:OMFT) it is decreased by $k' + 1$. Initially $\ell$ was m, but every iteration it is decreased by $k'$. Hence

$$q = n - k - (m - \ell_k) \qquad (23)$$

Now, $N - \{a_1, a_2, \ldots, a_q\} = \{1, 2, \ldots, k\} \cup \{n - m + \ell_k + 1, \ldots, n - 1, n\}$. Substituting in (22) we obtain (24) for $1 \leq k \leq r$

$$(S_{m-\ell_k})\beta + \sum_{j=1}^{\ell_k} f_{k+1-j}\ s_{m-\ell_k+j} = T_k + {}_{(n-m+\ell_k+1)}T \qquad (24)$$

where $T_j = \sum_{i=1}^{j} t_i$ and ${}_{(j)}T = \sum_{i=j}^{n} t_i$.

Equation (24) for $k = r$ is equation (25)

$$(S_{m-\ell_r})\beta + \sum_{j=1}^{\ell_r} f_{r+1-j}\ s_{m-\ell_r+j} = T_r + {}_{(n-m+\ell_r+1)}T \qquad (25)$$

Substituting $q = 0$ in equation (23), we obtain

$$r = n - (m - \ell_r) \qquad (26)$$

Substituting (26) in (25) we obtain

$$(S_{m-\ell_r})\beta + \sum_{j=1}^{\ell_r} f_{r+1-j}\ s_{m-\ell_r+j} = T_n. \qquad (27)$$

Using (27) we obtain equations for $r + 1 \leq k \leq n$

$$(S_{m-\ell_r-(k-r)})\beta + \sum_{j=1}^{k-r} s_{m-\ell_r-(k-r)+j}\, \beta + \sum_{j=1}^{\ell_r} f_{r+1-j}\, s_{m-\ell_r+j} = T_n.$$

Since $f_{k+1} \leq \beta, \ldots, f_n \leq \beta$.

$$(S_{m-\ell_r-(k-r)})\beta + \sum_{j=1}^{k-r} s_{m-\ell_r-(k-r)+j}\, f_{k+1-j} + \sum_{j=1}^{\ell_r} f_{r+1-j}\, s_{m-\ell_r+j} \leq T_n$$

$$(S_{m-\ell_r-(k-r)})\beta + \sum_{j=1}^{\ell_r+k-r} s_{m-\ell_r-(k-r)+j}\, f_{k+1-j} \leq T_n$$

$$\text{for } r + 1 \leq k \leq n. \qquad (28)$$

Equations and inequalities obtained using (24) and (28) for example 1.

| | |
|---|---|
| $k = 1,\ \ell_1 = 3$ | $f_1 s_1 = T_1$ |
| $k = 2,\ \ell_2 = 3$ | $f_1 s_2 + f_2 s_1 = T_2$ |
| $k = 3,\ \ell_3 = 3$ | $f_1 s_3 + f_2 s_2 + f_3 s_1 = T_3$ |
| $k = 4,\ \ell_4 = 3$ | $f_2 s_3 + f_3 s_2 + f_4 s_1 = T_4$ |
| $k = 5,\ \ell_5 = 2$ | $S_1\beta + f_4 s_3 + f_5 s_2 = T_5 + (8)^T$ |
| $k = 6,\ \ell_6 = 1$ | $S_2\beta + f_6 s_3 = T_6 + (7)^T$ |
| $k = 7$ | $S_1\beta + f_6 s_3 + f_7 s_2 \leq T_8$ |
| $k = 8$ | $f_6 s_3 + f_7 s_2 + f_8 s_1 \leq T_8$ |

In order to complete the proof of the theorem it is required to show that $mft(S) \leq mft(S')$ for any other feasible schedule $S'$ with $ft(S') \leq \beta$. The proof is by contradiction. Assume there is a schedule $S'$ with $ft(S') \leq \beta$ and $mft(S') < mft(S)$.

From lemma 2 it follows that $f'_1 \leq f'_2 \leq \ldots \leq f'_n$ , where $f'_i$ is the completion time for task $\tau_i$ in schedule $S'$. Now, we will obtain inequalities for schedule S'. Lemma 3 will be used $r$ times. For $1 \le i \le r$, the value of $k$ to be used in the lemma is $m - \ell_i + 1$ and $w$ is $i$. From 2) in lemma 3 we obtain

$$(s_1 + s_2 + \dots + s_{m-\ell_i})\beta + s_{m-\ell_i+1} f'_i + \dots + s_m f'_{i-\ell_i+1} \ge T_i + {}_{(n-m+\ell_i+1)}T$$

$$(S_{m-\ell_i})\beta + \sum_{j=1}^{\ell_i} f'_{i+1-j}\, s_{m-\ell_i+j} \ge T_i + {}_{(n-m+\ell_i+1)}T \quad \text{for} \quad 1 \le i \le r \qquad (29)$$

Inequalities $i$, for $r + 1 \le i \le n$, will be obtained using 1) in lemma 3, with $k = m - \ell_r + 1$ and $w = r$.

$$s_1 f'_n + s_2 f'_{n-1} + \dots + s_{m-\ell_r} f'_{n-m+\ell_r+1} + s_{m-\ell_r+1} f_r + \dots + s_m f_{r-\ell_r+1} \ge$$

$$T_r + {}_{(n-m+\ell_r+1)}T$$

From (26) we have that if $q = 0$ then $r = n - (m - \ell_r)$, so

$$s_1 f'_n + s_2 f'_{n-1} + \dots + s_{m-\ell_r} f'_{r+1} + s_{m-\ell_r+1} f_r + \dots + s_m f_{r-\ell_r+1} \ge T_n \qquad (30)$$

Now for $r + 1 \le i \le n$ as $ft(S') \le \beta$ we obtain from above

$$(S_{m-\ell_r-(i-r)})\beta + \sum_{j=1}^{\ell_r+i-r} s_{m-\ell_r-(i-r)+j}\, f_{i+1-j} \ge T_n \qquad (31)$$

Combining (24) and (28) with (30) and (31) we obtain

$$\sum_{j=1}^{i} a_{j,i} f_i \le \sum_{j=1}^{i} a_{j,i} f'_i \quad \text{for} \quad 1 \le i \le n$$

where $a_{j,i} \le a_{j+1,i}$ for $1 \le j \le i$.

Using lemma 4 ($\delta = S_m$) it follows that

$$\sum_{i=1}^{n} f_i \le \sum_{i=1}^{n} f'_i.$$

So, $mft(S') \geq mft(S)$, which contradicts our earlier assumption. Hence, algorithm Uβ:OMFT generates β:OMFT preemptive schedules for every system of $m \geq 1$ uniform processors and $n \geq 1$ independent tasks. □

Lemma 2 is stronger than theorem 3 given by Lawler and Labetoulle [LL]. From this lemma, it can be easily shown that there is an OMFT preemptive schedule in which jobs complete in nondecreasing order of their execution time. In addition, the finish time of this schedule is never greater than the one of any other OMFT preemptive schedule.

Lemma 2: Any schedule S for a uniform processor system can be transformed to a preemptive schedule S' with the following properties:

i) $f'_1 \leq f'_2 \leq \dots \leq f'_n$

ii) $ft(S') \leq ft(S)$

iii) $mft(S') \leq mft(S)$.

Proof: Properties i) and iii) follow from theorem 3 in [LL]. ii) follows from the observation that every time two jobs are swapped, the length of the schedule is never modified. □

Lemma 3: Given any β:OMFT preemptive schedule S' (with $f'_j = 0$ for $j \leq 0$ and $f'_1 \leq f'_2 \leq \dots \leq f'_n$) for m uniform machines, some k in [1; m] and a task index w in [1; n - k + 1]. The following inequalities hold:

1) $s_1 f'_n + s_2 f'_{n-1} + \dots + s_{k-1} f'_{n-k+2} + s_k f'_w + \dots + s_m f'_{w-m+k} \geq T_w + {}_{(n-k+2)}T$ and

2) $(s_1 + \dots + s_{k-1})\beta + s_k f'_w + \dots + s_m f'_{w-m+k} \geq T_w + {}_{(n-k+2)}T$.

Proof: First we prove 1). $S'$ is represented in figure 2. Note that $f'_{w-m+k}$ does not imply that $\tau_{w-m+k}$ will terminate on $P_m$, it indicates that $\tau_{w-m+k}$ terminates at that time.

Let R represent the shaded area of the schedule represented in figure 3. Now

$$s_1 f'_n + s_2 f'_{n-1} + \dots + s_{k-1} f'_{n-k+2} + s_k f'_w + \dots + s_{m-1} f'_{w-m+k+1} + s_m f'_{w-m+k}$$

$$= T_w + {}_{(n-k+2)}T + X + \sum_{i=w+1}^{n-k+1} \delta_i - \sum_{i=1}^{w} \rho_i - \sum_{i=n-k+2}^{n} \rho_i$$

where, X is the total processing capability of the idle time region inside R(fig. 2),

$\rho_i$ total processing time $\tau_i$ is scheduled outside R, for $i = 1, 2, \dots, w, n-k+2, \dots, n$.

and $\delta_i$ total processing time $\tau_i$ is scheduled inside R, for $w + 1 \le i \le n - k + 1$.

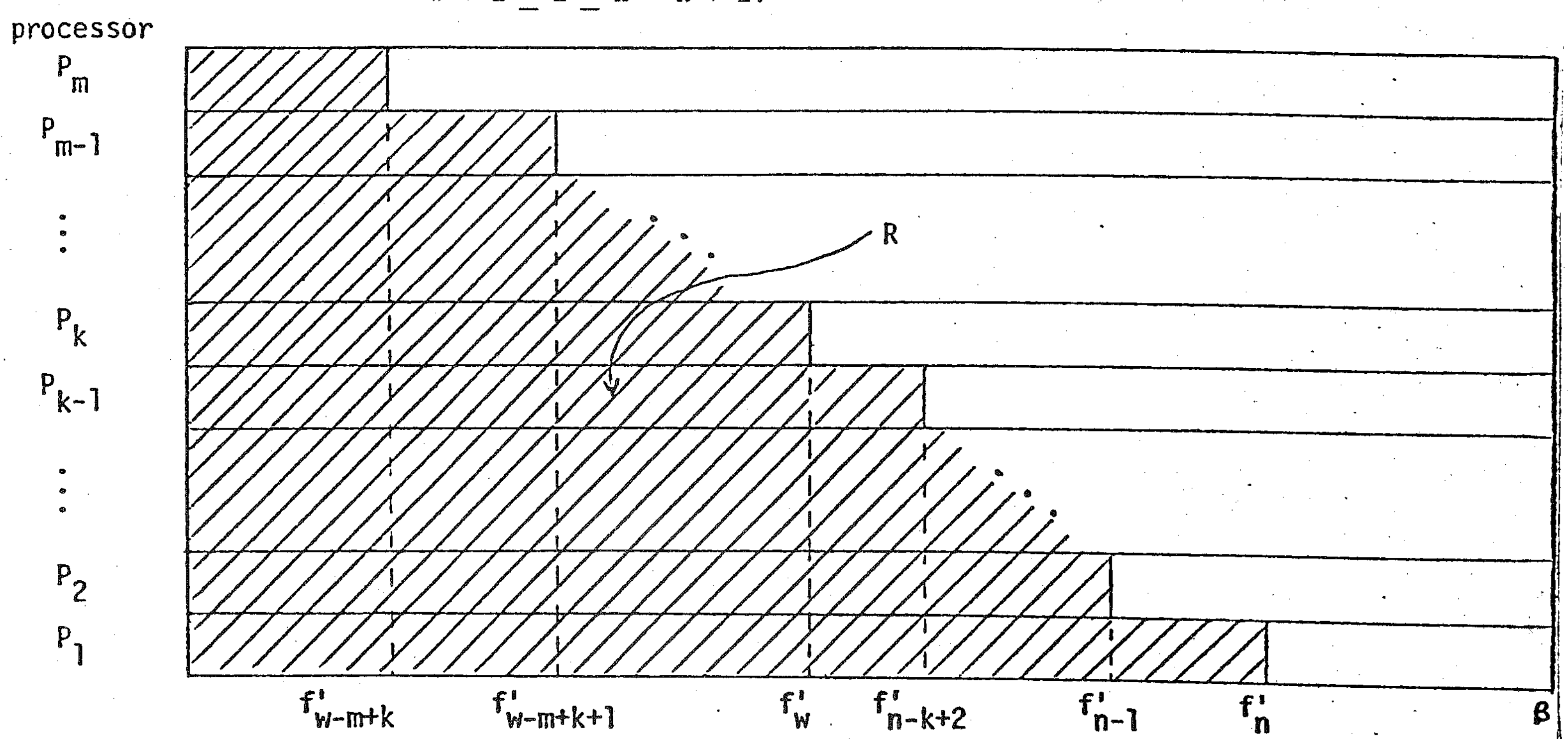


Figure 2: Schedule $S'$. In case $k = 1$, $f'_n, \dots, f'_{n-k+2}$ are not included. R is the shaded region.

To prove 1) it is required to show that

$$X + \sum_{i=2+1}^{n-k+1} \delta_i \geq \sum_{i=1}^{w} \rho_i + \sum_{i=n-k+2}^{n} \rho_i \qquad \text{a)}$$

This can be shown, if we prove that a) holds when we consider any time interval $\Delta$ in which all processors either execute one task or are idle, throughout the interval. There are two cases:

i) $\rho_i$ is zero in this interval if either all machines are inside $R$ or outside $R$, so a) holds.

ii) If $\ell$ of the machines are inside $R$, then only $\ell$ tasks in $\tau_{w-m+k}, \ldots, \tau_w, \tau_{n-k+2}, \ldots, \tau_n$ have not yet been completed. Clearly these $\ell$ tasks are the only tasks that can contribute to $\rho$ in this interval. But the speed of the machines inside $R$ is $\geq$ than the speed of the machines outside $R$, so the total contribution from the $\rho$'s is $\leq$ than the contribution of idle time and $\delta$'s. Then a) holds for the interval.

Then it follows that 1) holds for any schedule $S'$.

To prove 2), we use 1) and the restriction $ft(S') \leq \beta$. So,

$$(s_1 + \ldots + s_{k-1})\beta + s_k f'_w + \ldots + s_{m-1} f'_{w-m+k+1} + s_m f'_{w-m+k} \geq T_w + {}_{(n-k+2)}T.$$

This completes the proof of the lemma. □

Lemma 4: Given that

$$\sum_{j=1}^{i} a_{j',i} f_i \leq \sum_{j=1}^{i} a_{j',i} f'_i \quad \text{for} \quad 1 \leq i \leq n$$

where $a_{j,i} \leq a_{j+1,i}$ for $1 \leq j < i$, $\delta > 0$, $a_{j,i} \geq 0$, $a_{i,i} \geq 0$, $f_i \geq 0$ and $f'_i \geq 0$. Then $\sum_{j=1}^{n} f_i \leq \sum_{j=1}^{n} f'_i$.

Proof: Let $x_n = \delta/a_{n,n}$ and for $i = n - 1, \ldots, 1$

$$x_i = \frac{\delta - \sum_{j=i+1}^{n} a_{j,i} x_i}{a_{i,i}}$$

From this definition it can be easily shown that $x_i \geq 0$ for $1 \leq i \leq n$.

Multiplying the $x_i$'s by the inequalities we obtain

$$\sum_{i=1}^{n} x_i \sum_{j=1}^{i} a_{j,i} f_i \leq \sum_{i=1}^{n} x_i \sum_{j=1}^{i} a_{j,i} f'_i$$

Rearranging terms and eliminating the $x_i$'s we obtain

$$\delta \sum_{i=1}^{n} f_i \leq \delta \sum_{i=1}^{n} f'_i$$

As $\delta > 0$ then $\sum_{i=1}^{n} f_i \leq \sum_{i=1}^{n} f'_i$. □

Theorem 2: The time complexity for algorithm Uβ:OMFT is $O(nm)$.

Proof: Steps 1-3 take time $O(n + m)$. Since $q$ is decreased by at least one each time loop 4-22 is executed, it follows that loop 4-22 is executed at most $n$ times. Steps 5-7 can be easily implemented to take $O(m)$ time. Hence the overall time for steps 5-7 is $O(nm)$. Loop 9-15 is executed at most $m - 1$ times as each time $\ell$ is decreased by one and $\ell$ will never be smaller than one (line 5-6). Each time the loop takes times $O(n + m)$. Therefore the overall contribution of loop 9-15 is $O(nm)$. Using similar arguments it can easily be shown that the contribution of lines 8 and 16-21 take overall time $O(nm + m^2)$. Hence, the overall time complexity for algorithm Uβ:OMFT is $O(nm)$. □

<u>Theorem 3</u>: The maximum number of preemptions introduced by algorithm Uβ:OMFT is $0(nm)$.

<u>Proof</u>: Tasks are assigned in lines 11 and 16. Each DPN consists initially of at most $m$ blocks. If $k' = 0$, then $\tau_{a_q}$ will be scheduled with at most $m - 1$ preemptions. If $k' \neq 0$, then the total number in blocks in all the unused DPN's increases by at most one each time a task is scheduled. As $k' < m$, it follows that no task will be scheduled with more than $2m$ preemptions. Hence the total number of preemptions is $0(nm)$. □

## III. Conclusion

We have presented an algorithm to construct β:OMFT preemptive schedules for n independent tasks on m identical machines. The algorithm is of time complexity 0(nm) and introduces m - 1 preemptions. When β is large and the speed of all the processors is identical the algorithm reduces to the well known SPT rule. When β is large the algorithm reduces to a generalization of the SPT rule for uniform processors. The rules construct OMFT preemptive schedules for uniform processor systems. When the speed of the machines is the same the algorithm reduces to the one in [G2] to construct β:OMFT preemptive schedules for identical processor systems. It can be easily shown that the same type of algorithm will not construct OMFT preemptive schedules when there are two or more deadlines to be met by the tasks.

## Acknowledgements

The author is grateful to Professor John Bruno for his valuable comments and suggestions on earlier versions of this paper.

References


[BCS] J. Bruno, E. G. Coffman, Jr. and R. Sethi, "Scheduling independent tasks to reduce mean finishing time," Comm. ACM 17 (1974), 382-387.

[C] E. G. Coffman, Jr. (ed.) Computer and Job/Shop Scheduling Theory, 42-48, John Wiley & Sons, New York (1975).

[CMM] R. W. Conway, W. L. Maxwell, and L. W. Miller, "Theory of scheduling," Addison-Wesley, Reading, Mass., (1967).

[G1] T. Gonzalez, "A note on open shop preemptive schedules," TR-24 The Pennsylvania State University, Dec. 1976.

[G2] T. Gonzalez, "Minimizing the Mean and Maximum Finishing Time on Identical Processors," TR-15-78, The Pennsylvania State University, Sept. 1978.

[GS1] T. Gonzalez and S. Sahni, "Preemptive scheduling of uniform processor systems," JACM Vol. 25, No. 1, Jan. 1978.

[GS2] T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time," JACM Vol. 23, No. 4, Oct. 1976, 665-679.

[GS3] T. Gonzalez and S. Sahni, "Flowshop and jobshop schedules: Complexity and Approximations," JORSA Vol. 25, No. 1, Jan.-Feb. 1978, 36-52.

[HLS] E. C. Horvath, S. Lam and R. Sethi, "A level algorithm for preemptive scheduling," JACM Vol. 24, No. 1, Jan. 1977, 32-43.

[LL] E. L. Lawler and J. Labetoulle, "Scheduling of parallel machines with preemptions," IRIA, Rocquencourt, France.

[LY] J. W. S. Liu and A. Yang, "Optimal scheduling of independent tasks on heterogeneous computing systems," 1974 ACM National Conference, 38-45.

[Mc] R. McNaughton, "Scheduling with deadlines and loss functions," Management Science, 12 7 (1959), 1-12.

[MC1] R. R. Muntz and E. G. Coffman, Jr., "Preemptive scheduling of real time tasks on multiprocessor systems," JACM, 17, 2 (1970), 324-338.

[MC2] R. R. Muntz and E. G. Coffman, Jr., "Optimal preemptive scheduling on two-processor systems," IEEE Transactions on Computers, C-18, 11 (1969(, 1014-1020.

[SG] S. Sahni and T. Gonzalez, "Preemptive scheduling of two unrelated machines," Univ. of Minnesota, Nov. 1976.

[U] J. D. Ullman, "NP-complete scheduling problems," J. Computer and Systems Sciences, 10, 3 (June 1975), 384-393.

## Appendix

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $P_{i_\ell}$ | | $DPO_1$ | $DPO_2 \ldots DPO_{\ell-j}$ | $DPO_{\ell-j+1}$ | $DPO_{\ell-j+2} \cdots DPO_{\ell-2}$ | $DPO_{\ell-1}$ | $DPO_\ell$ |
| $P_{i_{\ell-1}}$ | | | $DPO_1 \ldots DPO_{\ell-j-1}$ | $DPO_{\ell-j}$ | $DPO_{\ell-j+1} \cdots DPO_{\ell-3}$ | $DPO_{\ell-2}$ | $DPO_{\ell-1}$ |
| ⋮ | | | | | | | |
| $P_{i_j}$ | | | | $DPO_1$ | $DPO_2 \ldots DPO_{j-2}$ | $DPO_{j-1}$ | $DPO_j$ |
| $P_{i_{j-1}}$ | | | | | $DPO_1 \ldots DPO_{j-3}$ | $DPO_{j-2}$ | $DPO_{j-1}$ |
| ⋮ | | | | | | | |
| $P_{i_2}$ | | | | | | $DPO_1$ | $DPO_2$ |
| $P_{i_1}$ | | | | | | | $DPO_1$ |

$0 \quad \mu_{i_\ell} \quad \mu_{i_{\ell-1}} \quad \ldots \quad \mu_{i_j} \quad \mu_{i_{j-1}} \quad \ldots \quad \mu_{i_2} \quad \mu_{i_1} \quad \beta$

Figure 3a: DPO's

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $P_{i_\ell}$ | | $DPN_1$ | $DPN_2 \ldots DPN_{\ell-j}$ | $DPN_{\ell-j+1}$ | $DPN_{\ell-j+2} \cdots DPN_{\ell-2}$ | $DPN_{\ell-1}$ | $DPN_\ell$ | $DPN_{\ell+1}$ |
| $P_{i_{\ell-1}}$ | | | $DPN_1 \ldots DPN_{\ell-j-1}$ | $DPN_{\ell-j}$ | $DPN_{\ell-j+1} \cdots DPN_{\ell-3}$ | $DPN_{\ell-2}$ | $DPN_{\ell-1}$ | $DPN_\ell$ |
| ⋮ | | | | | | | | |
| $P_{i_j}$ | | | | $DPN_1$ | $DPN_2 \ldots DPN_{j-2}$ | $DPN_{j-1}$ | $DPN_j$ | $DPN_{j+1}$ |
| $P_{i_{j-1}}$ | | | | | $DPN_1 \ldots DPN_{j-3}$ | $DPN_{j-1}$ | $DPN_{j-1}$ | $DPN_j$ |
| ⋮ | | | | | | | | |
| $P_{i_2}$ | | | | | | $DPN_1$ | $DPN_2$ | $DPN_3$ |
| $P_{i_1}$ | | | | | | | $DPN_1$ | $DPN_2$ |

$0 \quad \mu_{i_\ell} \quad \mu_{i_{\ell-1}} \quad \mu_{i_j} \quad \mu_{i_{j-1}} \quad \mu_{i_2} \quad \mu_{i_1} \quad \mu_{i_0} \quad \beta$

Figure 3b: DPN's