Lost

Minimizing the Mean and Maximum Finishing Time on Identical Processors[†]

Teofilo Gonzalez, August 1978
CS-78-15

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

Abstract: The problem of preemptively scheduling a set of n independent tasks on m identical 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 $m - 1$ preemptions.

Keywords: identical processors, preemptive schedules, OFT, OMFT, β:OMFT, OFT:OMFT, polynomial complexity.

[†]This research was supported in part by NSF grant MCS77-21092.

I. Introduction

There are $n \geq 1$ independent tasks to be scheduled on an identical processor system. Tasks shall be denoted by $\tau_1, \tau_2, \ldots, \tau_n$ and have execution time requirements $t_1 \leq t_2 \leq \cdots \leq t_n$. An identical processor system consists of $m \geq 1$ processors denoted by $P_1, P_2, \ldots, P_m$.

Let $w_i > 0$ be the weight given to task τ_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 $w_1 \geq w_2 \geq \cdots \geq w_n$. 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]. Let's just mention one. Consider the tasks as originating from different users, e.g., users requesting the execution of programs in a multiaccess, multiprocessor computer system. It is assumed that all processing units share a common block of storage, processors can execute different programs at the same time, the cost of interrupting the execution of a program is zero and all programs as well as data reside in main storage. Lets consider the system when several users are requesting the execution of their programs. The time a user will wait before its service request is satisfied is referred to as the <u>response time</u>. An Operating System could have been designed to assign processsing units to user programs in different ways, so as to optimize different cost functions. If the system uses an OFT scheduling policy, then the maximum response time is minimized. On the other hand, an OMFT scheduling policy will guarantee that the average response time is minimized. Known algorithms used to implement an OFT scheduling policy will not guarantee good average response time and the ones used for OMFT scheduling do not minimize the maximum response time. One way to compromise, is to use an OFT:OMFT scheduling policy. This will guarantee the maximum response time to be minimum and from the set of all possible ways this could be guaranteed, we select the one that will minimize the average response time. In general, a β:OMFT scheduling policy, will guarantee the maximum response time to be not greater than β and users are serviced in such a way that the average response time is minimized.

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 β:OMFT preemptive schedules for identical machines is presented. Special cases of this problem are the construction of OFT:OMFT and OMFT preemptive schedules.

For identical processors, McNaughton [Mc] presents a $0(n)$ 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 processor systems, i.e., when some processors are faster than others, [LY], [HLS] and [GS1] present polynomial time bounded algorithms to obtain OFT preemptive schedules.

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 $0(n \log n)$ time. For uniform processor systems, [LL] and [G2] present algorithms to obtain OMFT preemptive schedules. The problem of constructing OWMFT preemptive schedules for identical processors is NP-hard $(m \geq 2)$ [LL].

For nonpreemptive scheduling, [BCS] present approximation algorithms, in which they simultaneously minimize the ft and mft. These results are summarized in [C, pp. 42-49].

An algorithm to construct β:OMFT preemptive schedules for an identical processor system is presented in section II. The algorithm is of time complexity $0(nm)$ and introduces no more than $m - 1$ preemptions.

II. β:OMFT Preemptive Schedules for Identical Processors

An algorithm to obtain OMFT preemptive schedules for identical processors in which all tasks are required to complete by time β is presented. Note that $\beta \geq \max\{t_n, \Sigma t_i/m\}$ as otherwise no such schedule could be constructed.

β:OMFT preemptive schedules can be constructed 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 3 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 $0(nm)$ for this problem is presented and analyzed. The maximum number of preemptions introduced is $m - 1$.

Algorithm Iβ:OMFT considers task τ_i at step i. τ_i is scheduled 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. Algorithm Iβ:OMFT schedules task τ_i and procedure REARRANGE will find the minimum completion time (t) for task τ_i. In order to simplify the algorithm, REARRANGE will schedule all other tasks whose completion time is β in any other feasible schedule including the schedule so far constructed with τ_i scheduled to complete at time t.

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 μ_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 by REARRANGE, i.e., those processors for which μ was given the value β by procedure REARRANGE. Initially, all processors are of the first type. 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 τ_i is to be executed from time s to time f by processor P_j. Let f_i be the completion time for task τ_i in schedule Q. Assume $f_i = 0$ for $i \leq 0$.

```
    algorithm Iβ:OMFT (m, n, β, Q, t)
    //given m identical processors, the algorithm constructs a β:OMFT
      preemptive schedule for tasks $\tau_1, \tau_2, \ldots, \tau_n$ whose execution
      time requirements are $t_1 \leq t_2 \leq \cdots \leq t_n$. The schedule for pro-
      cessor $P_j$ is represented by $Q_j$. The kth tuple in $Q_j$ is of
      the form (i, s, f), indicating that $\tau_i$ is to be executed from time
      s to time f by processor $P_j$.//
    //initialize the processor schedules//
1     [$Q_j \leftarrow \emptyset$; $\mu_j \leftarrow 0$] for $1 \leq j \leq m$
    //set processor and task indices//.
2     $i_j \leftarrow j$ for $1 \leq j \leq m$
3     $a_j \leftarrow n - j + 1$ for $1 \leq j \leq n$
4     $\ell \leftarrow m$; $q \leftarrow n$;
    //schedule task $\tau_{a_q}$//
```

```
5    while $q \neq 0$ do
       //REARRANGE finds the minimum time $\mu_{i_1}$ such that if $\tau_{a_q}$ is assigned
         from time $\mu_{i_1}$ to $\mu_{i_1} + t_{a_q}$ then the remaining tasks can be
         scheduled to complete by time $\beta$.//
7      REARRANGE
       //assign $\tau_{a_q}$ to $P_{i_1}$//
8      $Q_{i_1} \leftarrow Q_{i_1} \| (a_q, \mu_{i_1}, \mu_{i_1} + t_{a_q})$ ; $\mu_{i_1} \leftarrow \mu_{i_1} + t_{a_q}$
       //rotate processors so that $\mu_{i_1} \leq \mu_{i_2} \leq \ldots \leq \mu_{i_\ell}$//
9      $(i_1, i_2, \ldots, i_\ell) \leftarrow (i_2, i_3, \ldots, i_\ell, i_1)$
       //eliminate $\tau_{a_q}$ from the set of tasks not yet been scheduled//
10     $q \leftarrow q - 1$
11   endwhile
12 end of algorithm IB:OMFT
```

If procedure REARRANGE does nothing, the reader will find it simple to verify that algorithm Iβ:OMFT constructs an SPT schedule ([C p. 14] and [CMM p. 26]). The first three iterations in example 1 show this effect.

At some point during the execution of algorithm Iβ:OMFT , all or some of the following properties will be true.

a1) Q is a schedule for tasks τ_k , $k \varepsilon N - A$

a2) $\mu_{i_1} \leq \mu_{i_2} \leq \ldots \leq \mu_{i_\ell} \leq \beta$

$\mu_k = \beta$ for $k \varepsilon M - I$

a3) $\sum_{j=1}^{k} (\beta - \mu_{i_j}) \geq \sum_{j=1}^{k} t_{a_j}$ for $1 \leq k \leq \min\{q, \ell - 1\}$

a4) $\sum_{j=1}^{m} \mu_j = \sum_{j \varepsilon N-A} t_j$ (Assume $\sum_{j \varepsilon \phi} t_j = 0$)

a5) $t_{a_k} \geq t_{a_{k+1}}$ for $1 \leq k < q$

a6) $a_i - 1 = a_{i+1}$ for $1 \le i < q$

$a_q = w + 1$, where w is the number of times loop 5-11 (Iβ:OMFT) has been executed ($a_0 = r + 1$, where r is the total number of times loop 5-11 was executed when the algorithm terminates. Assume $t_{n+1} \ge t_n$).

a7) $\mu_{i_\ell} - \mu_{i_1} \le t_{a_q}$

a8) $f_{a_q-1+j-\ell} = \mu_{i_j}$ for $1 < j \le \ell$ (Note that $f_i = 0$ for $i \le 0$).

a9) $f_{a_q-\ell} = \mu_{i_1}$

a10) $\ell > 0$

a11) $q > 0$

a12) $\sum_{j=1}^{k} (\beta - \mu_{i_{j+1}}) > \sum_{j=1}^{k} t_{a_j}$ for $1 \le k \le \min\{q - 1, \ell - 1\}$.

In lemma 1, we show that a1-a10 will hold true every time line 5 is executed. Lemma 2, completes the proof of lemma 1. In theorem 1, we show that algorithm Iβ:OMFT constructs β:OMFT preemptive schedules. The proof for this theorem uses lemma 1 together with lemma 3-5. Lemmas 3 and 4 study some properties of general preemptive schedules.

<u>Lemma 1</u>: At the beginning of each iteration a1-a10 will hold true.

<u>proof</u>: It is easy to verify that the lemma is true at the beginning of the first iteration. We now show that if a1-a10 hold true and $q \ne 0$ just before line 5 then after the execution of lines 6-11, a1-a10 will hold true.

In line 7, there is a call to procedure REARRANGE. Clearly, the conditions of lemma 2 are satisfied. Hence a1-a8 and a10-a12 hold true after line 7.

Let us show that just before line 8 is executed 1) and 2) hold true

1) $\mu_{i_1} + t_{a_q} \leq \beta$

There are two cases:

case i) $\ell = 1$

Substituting $\mu_k = \beta$ for $k \varepsilon M - \{i_1\}$ (a2) in $\sum_{j=1}^{m} \mu_j = \sum_{j \varepsilon N-A} t_j$ (a4), we obtain $\mu_{i_1} + (m-1)\beta = \sum_{j \varepsilon N-A} t_j$. Substituting the initial condition $\beta \geq_{(1)} T/m$, we get $\mu_{i_1} + \sum_{j \varepsilon A} t_j \leq \beta$. As $q > 0$ (a11) it follows that $\mu_{i_1} + t_{a_q} \leq \beta$.

case ii) $\ell > 1$

As $q > 0$ (a11), then $\beta - \mu_{i_1} \geq t_{a_1}$ (a3) and $t_{a_1} \geq t_{a_q}$ (a5). So, $\mu_{i_1} + t_{a_q} \leq \beta$.

This completes the proof of 1.

2) $t_{a_{q-1}} \geq t_{a_q}$ and $a_{q-1} - 1 = a_q$.

When $q > 1$, $t_{a_{q-1}} \geq t_{a_q}$ follow from a5 and $a_{q-1} - 1 = a_q$ follow from a6. If $q = 1$, then it is the last iteration. By definition, $a_0 = r + 1$. $a_1 = r$ as a6 holds true. Hence $a_0 - 1 = a_1$. From the initial conditions we have that $t_1 \leq t_2 \leq \ldots \leq t_n$. By definition $t_n \leq t_{n+1}$. $r \leq n$. Consequently $t_{a_{q-1}} \geq t_{a_q}$.

In order to complete the proof of the lemma, it is required to show that a1-a10 will hold true at the end of the iteration. The proof is given in a)-g).

a) a5 and a10 are obviously true at the end of the iteration.

b) $\sum_{j=1}^{m} \mu_j = \sum_{j \varepsilon N-A} t_j$, $\ell > 0$, $q > 0$ and Q is a schedule for τ_k, $k \varepsilon N-A$ (a4, a10, a11 and a1). In line 8, τ_{a_q} is assigned for t_{a_q} time units to processor P_{i_1}. Hence $\sum_{j=1}^{m} \mu_j = \sum_{j \varepsilon N-\{a_1, \ldots, a_{q-1}\}} t_j$ and Q is a schedule for τ_k, $k\epsilon N - \{a_1, \ldots, a_{q-1}\}$. q is decreased in line 10. Consequently a1 and a4 are true at the end of the iteration.

c)* $\sum_{j=1}^{k} (\beta - \mu_{i_{j+1}}) > \sum_{j=1}^{k} t_{a_j}$ for $1 \leq k \leq \min\{q - 1, \ell - 1\}$ (a12). Line 8 does not modify μ_{i_k} for $k > 1$. In line 9, processors are rotated and q is decreased in line 10. Hence $\sum_{j=1}^{k} (\beta - \mu_{i_j}) \geq \sum_{j=1}^{k} t_{a_j}$ for $1 \leq k \leq \min\{q, \ell - 1\}$. Consequently a3 is true after line 10.

d) $\mu_{i_1} + t_{a_q} \leq \beta$, $\mu_{i_\ell} - \mu_{i_1} \leq t_{a_q}$, $\ell > 0$, $\mu_{i_1} \leq \mu_{i_2} \leq \cdots \leq \mu_{i_\ell} \leq \beta$ and $\mu_k = \beta$ for $k\epsilon M - I$ (1, a7, a10, and a2). In line 8, μ_{i_1} is set to $\mu_{i_1} + t_{a_q}$. Hence $\mu_{i_2} \leq \cdots \leq \mu_{i_\ell} \leq \mu_{i_1} \leq \beta$ and $\mu_k = \beta$ for $k\epsilon M - I$. Processors are rotated (line 9). Consequently a2 is true at the end of the iteration.

e) $a_{q-1} - 1 = a_q$, $a_i - 1 = a_{i+1}$ for $1 \leq i < q$ and $a_q = w + 1$ (2, a6). Lines 8-9 have no effects. Line 10 decreases q. Hence, $a_i - 1 = a_{i+1}$ for $1 \leq i < q$ and $a_q = w + 2$. At this point loop 5-11 has been completed and we may think of w as being increased. So, a6 is true at the end of the iteration.

f) It should be clear that a7 is true after line 10 when $\ell = 1$. For $\ell > 1$, we have that $\mu_{i_2} \geq \mu_{i_1}$ and $t_{a_{q-1}} \geq t_{a_q}$ (a2, 2). Adding both inequalities, $\mu_{i_2} + t_{a_{q-1}} \geq \mu_{i_1} + t_{a_q}$. After line 8, this becomes $\mu_{i_2} + t_{a_{q-1}} \geq \mu_{i_1}$. In line 9 processors are rotated and q is decreased in line 10. Hence $\mu_{i_1} + t_{a_q} \geq \mu_{i_\ell}$. Consequently a7 holds true after line 10.

g) $f_{a_q - 1 + j - \ell} = \mu_{i_j}$ for $1 < j \leq \ell$ and $a_{q-1} - 1 = a_q$ (a8 and 2). In line 8, f_{a_q} is set to μ_{i_1}. Processors are rotated in line 9. Hence $f_{a_{q-1} - 1 + j - \ell} = \mu_{i_j}$ for $1 \leq j \leq \ell$. In line 10, q is decreased. Consequently a8 and a9 hold true after line 10.

Hence a1-a10 will hold true at the end of each iteration. This completes the proof of the lemma. □

*If $\ell = 1$ or $q = 1$ just after line 7, then a3 holds true after line 10. So assume $\ell > 1$ and $q > 1$ just after line 7.

Procedure REARRANGE is used by algorithm $I\beta$:OMFT. REARRANGE is constructed in such a way that if a1-a11 hold true before it is called, then a1-a8 and a10-a12 will hold true just before the procedure terminates.

```
    procedure REARRANGE
    //all variables are global//
    //given that a1-a11 hold true just before a call to REARRANGE, the
      procedure schedules a subset of tasks in such a way that a1-a8
      and a10-a12 will hold true just before the procedure terminates//
    //the loop is not executed when  $q = 1$  or  $\ell = 1$//
1   while $\sum_{z=1}^{k} (\beta - \mu_{i_{z+1}}) \leq \sum_{z=1}^{k} t_{a_z}$ for some $k$ in $[1; \min\{q - 1, \ell - 1\}]$ do
2       Let $p'$ be the smallest integer such that $(\beta - \mu_{i_{z+1}}) > t_{a_z}$ for
        $1 \leq z < p'$ and $(\beta - \mu_{i_{p'+1}}) \leq t_{a_{p'}}$
        //schedule $\tau_{a_{p'}}$ on $P_{i_{p'+1}}$ and $P_{i_{p'}}$//
3       $Q_{i_{p'}} \leftarrow Q_{i_{p'}} \| (a_{p'}, \overset{\mu_{i_{p'}}}{\smile} \mu_{i_{p'}} \overset{+}{\cancel{,}}\, t_{a_{p'}} - (\beta - \mu_{i_{p'+1}}))$
4       $\mu_{i_{p'}} \leftarrow \mu_{i_{p'}} + t_{a_{p'}} - (\beta - \mu_{i_{p'+1}})$
5       $Q_{i_{p'+1}} \leftarrow Q_{i_{p'+1}} \| (a_{p'}, \mu_{i_{p'+1}}, \beta)$
6       $\mu_{i_{p'+1}} \leftarrow \beta$
        //eliminate $P_{i_{p'+1}}$ and $\tau_{a_{p'}}$//
7       $(i_1, i_2, \ldots, i_{\ell-1}) \leftarrow (i_1, \ldots, i_{p'}, i_{p'+2}, \ldots, i_\ell)$
8       $\ell \leftarrow \ell - 1$
9       $(a_1, a_2, \ldots, a_{q-1}) \leftarrow (a_1, \ldots, a_{p'-1}, a_{p'+1}, \ldots, a_q)$
10      $q \leftarrow q - 1$
11  endwhile
12 end of procedure REARRANGE
```

Lemma 2: If a1-a11 hold true just before a call to REARRANGE, then a1-a8 and a10-a12 will hold true just before the procedure terminates.

proof: Let's consider the case where loop 2-10 is not executed. By assumption, a1-a8 and a10-a11 hold true. a12 follow from the test in line 1. Hence a1-a8 and a10-a12 will hold true just before the procedure terminates.

The more interesting case is when loop 2-10 is executed. At the beginning of the first iteration a1-a8 and a10-a11 hold true. In order to complete the proof of the lemma, we prove the following three parts:

1) If a1-a5, a7 and a10-a11 hold true at the beginning of some iteration, then a1-a5, a7 and a10-a11 will hold true at the end of the iteration.

2) If a6 and a8 hold true at the beginning of some iteration, then after several iterations a6 and a8 will hold true.

3) a12 hold true at the end of the last iteration.

Let us now prove each part separately.

1) If a1-a5, a7 and a10-a11 hold true at the beginning of some iteration, then a1-a5, a7 and a10-a11 will hold true at the end of the iteration.

First, let us prove that just before line 3

i) $1 \leq p' \leq \min\{\ell - 1, q - 1\}$

and ii) $\mu_{i_{p'}} + t_{a_{p'}} - (\beta - \mu_{i_{p'+1}}) \leq \mu_{i_{p'+1}}$.

i) $\underline{1 \leq p' \leq \min\{\ell - 1, q - 1\}}$

The proof follows directly from lines 1 and 2.

ii) $\underline{\mu_{i_{p'}} + t_{a_{p'}} - (\beta - \mu_{i_{p'+1}}) \leq \mu_{i_{p'+1}}}$

This is equivalent to $\mu_{i_{p'}} + t_{a_{p'}} \leq \beta$. There are two cases:

<u>case ii.1: $p' > 1$</u>

By assumption $t_{a_{p'-1}} \geq t_{a_{p'}}$ (a5). From line 2 we have that $\beta - \mu_{i_{p'}} > t_{a_{p'-1}}$. Hence $\mu_{i_{p'}} + t_{a_{p'}} < \beta$.

<u>case ii.2: $p' = 1$</u>

$\ell > 1$ as otherwise the loop would not be executed (see line 1). So, $\beta - \mu_{i_1} \geq t_{a_1}$ (a3). As $p' = 1$, then $\mu_{i_{p'}} + t_{a_{p'}} \leq \beta$.

We now prove that a1-a5, a7 and a10-a11 hold true at the end of the iteration. The proof is in parts.

a) $1 \leq p' \leq \min\{\ell - 1, q - 1\}$, $\mu_{i_{p'}} + t_{a_{p'}} - (\beta - \mu_{i_{p'+1}}) \leq \mu_{i_{p'+1}}$ and Q is a schedule for τ_k, $k \in N - A$ (i, ii and a1). In lines 3-6, $\tau_{a_{p'}}$ is assigned for $t_{a_{p'}}$ time units, part to $P_{i_{p'}}$ and part to $P_{i_{p'+1}}$. Hence Q is a schedule for τ_k, $k \in N - \{a_1, \ldots, a_{p'-1}, a_{p'+1}, \ldots, a_{q-1}\}$. Lines 7-8 have no effect on Q. Lines 9-10 eliminate $\tau_{a_{p'}}$ and q is decreased. Hence Q is a schedule for τ_k, $k \in N - A$. Consequently a1 holds true at the end of the iteration.

b) $\mu_{i_{p'+1}} \geq \mu_{i_{p'}} + t_{a_{p'}} - (\beta - \mu_{i_{p'+1}})$, $\mu_{i_1} \leq \mu_{i_2} \leq \ldots \mu_{i_\ell} \leq \beta$ and $\mu_k = \beta$ for $k \in M - I$ (ii and a2). In line 3, $\mu_{i_{p'}}$ is set to $\mu_{i_{p'}} + t_{a_{p'}} - (\beta - \mu_{i_{p'+1}})$. Hence $\mu_{i_1} \leq \ldots \leq \mu_{i_{p'}} \leq \mu_{i_{p'+1}} \leq \mu_{i_{p'+2}} \leq \ldots \leq \mu_{i_\ell} \leq \beta$ and $\mu_k = \beta$ for $k \in M - I$. In lines 5-6 $\mu_{i_{p'+1}}$ is set to β. So $\mu_{i_1} \leq \ldots \leq \mu_{i_{p'}} \leq \mu_{i_{p'+2}} \leq \ldots \leq \beta$ and $\mu_k = \beta$ for $k \in M - (I - \{i_{p'+1}\})$. This becomes $\mu_{i_1} \leq \mu_{i_2} \leq \ldots \leq \mu_{i_\ell} \leq \beta$ and $\mu_k = \beta$ for $k \in M - I$ after line 8. Consequently a2 is true after line 10.

c) $\sum_{j=1}^{k} (\beta - \mu_{i_j}) \geq \sum_{j=1}^{k} t_{a_j}$ for $1 \leq k \leq \min\{q, \ell - 1\}$ (a3). This can be decomposed into

$$\sum_{j=1}^{k} (\beta - \mu_{i_j}) \geq \sum_{j=1}^{k} t_{a_j} \quad \text{for} \quad 1 \leq k \leq p' - 1 \tag{1}$$

and
$$\sum_{j=1}^{k} (\beta - \mu_{i_j}) \geq \sum_{j=1}^{k} t_{a_j} \quad \text{for} \quad p' + 1 \leq k \leq \min\{q, \ell - 1\} \tag{2}$$

Let μ' be the new value for μ after line 6. Lines 3-6 will set μ' as follows: $(\mu'_{i_1}, \mu'_{i_2}, \ldots, \mu'_{i_{p'}}, \mu'_{i_{p'+1}}, \ldots, \mu'_{i_\ell}) \leftarrow (\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_{p'}} + t_{a_{p'}} - (\beta - \mu_{i_{p'+1}}), \beta, \ldots, \mu_{i_\ell})$. Substituting in (1) and (2) we obtain (3) and (4)

$$\sum_{j=1}^{k} (\beta - \mu'_{i_j}) \geq \sum_{j=1}^{k} t_{a_j} \quad \text{for} \quad 1 \leq k \leq p' - 1 \tag{3}$$

and
$$\sum_{j=1}^{p'} (\beta - \mu'_{i_j}) + \sum_{j=p'+2}^{k} (\beta - \mu'_{i_j}) \geq \sum_{j=1}^{p'-1} t_{a_j} + \sum_{j=p'+1}^{k} t_{a_j}$$
$$\text{for} \quad p' + 1 \leq k \leq \min\{q, \ell - 1\} \tag{4}$$

After lines 7-10, these inequalities become

$$\sum_{j=1}^{k} (\beta - \mu'_{i_j}) \geq \sum_{j=1}^{k} t_{a_j} \quad \text{for} \quad 1 \leq k \leq p' - 1$$

and
$$\sum_{j=1}^{k} (\beta - \mu'_{i_j}) \geq \sum_{j=1}^{k} t_{a_j} \quad \text{for} \quad p' \leq k \leq \min\{q, \ell - 1\}.$$

So, a3 hold true after line 10.

d) $\sum_{j=1}^{m} \mu_j = \sum_{j \in N-A} t_j$ (a4). In lines 3-6, $\mu_{i_{p'}}$ and $\mu_{i_{p'+1}}$ are increased. The total increment is $t_{a_{p'}}$. In lines 9-10 $a_{p'}$ is eliminated and q is decreased. Hence $\sum_{j=1}^{m} \mu_j = \sum_{j \in N-A} t_j$. Consequently a4 holds true after line 10.

e) The proof for a5 is straight forward.

f) The proof for a7 is in two parts:

<u>case f.1 $p' < \ell - 1$</u>:

$$\mu_{i_\ell} - \mu_{i_1} \le t_{a_q} \quad \text{and} \quad 1 \le p' \le \min\{q - 1, \ell - 1\}$$

(a7 and i). μ_{i_ℓ} is not modified and μ_{i_1} will never be decreased. Hence $\mu_{i_\ell} - \mu_{i_1} \le t_{a_q}$ after line 6. As $1 \le p' \le \min\{q - 1, \ell - 1\}$, then after lines 7-10, $\mu_{i_\ell} - \mu_{i_1} \le t_{a_q}$.

<u>case f.2 $p' = \ell - 1$</u>:

$$\mu_{i_\ell} - \mu_{i_1} \le t_{a_q}, \ \mu_{i_{\ell-1}} \le \mu_{i_\ell} \quad \text{and}$$

$\mu_{i_{p'}} + t_{a_{p'}} - (\beta - \mu_{i_{p'+1}}) \le \mu_{i_{p'+1}}$ (a7, a2 and ii). After line 4, we have $\mu_{i_\ell} - \mu_{i_1} \le t_{a_q}$ and $\mu_{i_{\ell-1}} \le \mu_{i_\ell}$. Consequently $\mu_{i_{\ell-1}} - \mu_{i_1} \le t_{a_q}$. Lines 5-6 do not modify $\mu_{i_{\ell-1}}$ or μ_{i_1}. After lines 7-8 $\mu_{i_\ell} - \mu_{i_1} \le t_{a_q}$. As $1 \le p' \le \min\{q - 1, \ell - 1\}$, then after lines 9-10 $\mu_{i_\ell} - \mu_{i_1} \le t_{a_q}$.

Hence a7 holds true after line 10.

g) The proof for a10 and a11 follows from the fact that $q > 1$ and $\ell > 1$ in line 1 as otherwise the loop would not be executed.

This completes the proof of 1. Let us prove 2.

2) If a6 and a8 hold true at the beginning of some iteration, then after several iterations a6 and a8 will hold true.

The proof for this part is different than the one for 1). The reader will find it simple to verify that a6 and a8 will not hold true every iteration. However, after several iterations a6 and a8 will hold true. First of all, we show that c6, c8 and c12 hold true every iteration. Then we show that after k' iterations a6 and a8 will hold true. Let k' be the minimum value of k for which the condition in line 1 is true when a6 and a8 hold true just before line 1 is executed. Let x be the number of times loop 2-10 has been executed after k' was set. Let

c6) $a_j - 1 = a_{j+1}$ for $k' + 1 - x \leq j < q$

$a_q = w + 1$

c8) $f_{a_q - 1+j-\ell} = \mu_{i_j}$ for $k' + 2 - x \leq j \leq \ell$

c12) $\sum_{j=1}^{k'-x} (\beta - \mu_{i_{j+1}}) \leq \sum_{j=1}^{k'-x} t_{a_j}$.

By assumption c6, c8 and c12 hold true when $x = 0$. Suppose now $x < k'$ and c6, c8 and c12 hold true. We now show that the loop will be executed and c6, c8 and c12 will hold true at the end of the iteration. Clearly c12 guarantees that the loop will be executed. Now, in lines 3-6, $\tau_{a_{p'}}$ is assigned to $P_{i_{p'}}$ and $P_{i_{p'+1}}$. As $1 \leq p' \leq k' - x$, then c6 and c8 will remain true. After lines 7-8 we have that

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

$a_q = w + 1$

and $f_{a_q - 1+j-\ell} = \mu_{i_j}$ for $k' + 1 - x \leq j \leq \ell$.

After lines 9-10 this becomes

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

$a_q = w + 1$

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

Clearly, the loop has terminated and x should be increased. Hence c6 and c8 hold true at the end of the iteration. Let us now show that c12 will hold true at the end of the iteration. After line 2 we have that

$$\sum_{j=1}^{k'-x} (\beta - \mu_{i_{j+1}}) \leq \sum_{j=1}^{k'-x} t_{a_j}.$$

This can be decomposed into

$$\sum_{j=1}^{p'-2} (\beta - \mu_{i_{j+1}}) + (\beta - \mu_{i_{p'}}) + (\beta - \mu_{i_{p'+1}}) + \sum_{j=p'+1}^{k'-x} (\beta - \mu_{i_{j+1}}) \le$$

$$\sum_{j=1}^{p'-1} t_{a_j} + t_{a_{p'}} + \sum_{j=p'+1}^{k'-x} t_{a_j}$$

as $1 \le p' \le k' - x$. After line 4 this inequality becomes

$$\sum_{j=1}^{p'-2} (\beta - \mu_{i_{j+1}}) + (\beta - \mu_{i_{p'}}) + \sum_{j=p'+1}^{k'-x} (\beta - \mu_{i_{j+1}}) \le \sum_{j=1}^{p'-1} t_{a_j} + \sum_{j=p'+1}^{k'-x} t_{a_j}.$$

Clearly lines 5-6 have no effect and after lines 7-8 this inequality becomes

$$\sum_{j=1}^{k'-x-1} (\beta - \mu_{i_{j+1}}) \le \sum_{j=1}^{p'-1} t_{a_j} + \sum_{j=p'+1}^{k'-x} t_{a_j}.$$

Lines 9-10 will modify the inequality to

$$\sum_{j=1}^{k'-x-1} (\beta - \mu_{i_{j+1}}) \le \sum_{j=1}^{k'-x-1} t_{a_j}.$$

As the loop has terminated we have that x should be increased. Hence c12 holds true at the end of the loop.

Clearly c6, c8 and c12 hold true when $x = 0, 1, \ldots, k'$. Consequently a6 and a8 hold true after the k' th iteration. If the test in line 1 is true, then we repeat the same arguments (find a new k') until we reach a point when the condition is false. Such a point will be reached as ℓ is decreased each time by 1 and when $\ell = 1$ the condition in line 1 is false. Hence just before termination a6 and a8 will hold true. This completes the proof of part 2. We now prove part 3.

3) a12 will hold true at the end of the last iteration.

The proof follows directly from the test in line 1.

This completes the proof for parts 1), 2), 3) and the lemma. □

Example 1: For 8 tasks with execution times $t_1 = 2$, $t_2 = 4$, $t_3 = 4$, $t_4 = 6$, $t_5 = 8$, $t_6 = 10$, $t_7 = 12$, $t_8 = 14$; 4 identical processors and a deadline $\beta = 15$, algorithm Iβ:OMFT constructs a β:OMFT preemptive schedule as follows:

The different values for the variables are given in the following table.

$(a_1, a_2, \ldots, a_q)$	(q)	$(i_1, i_2, \ldots, i_\ell)$	(ℓ)	$(\mu_1, \mu_2, \ldots, \mu_m)$	schedule*	STEP
(8,7,6,5,4,3,2,1)	(8)	(1,2,3,4)	(4)	(0,0,0,0)	$Q_1 = Q_2 = Q_3 = \emptyset$	initial conditions
				SAME		after first call to REARRANGE
(8,7,6,5,4,3,2)	(7)	(2,3,4,1)	(4)	(2,0,0,0)	$Q_1 = [(1,0,2)]$ $Q_2 = Q_3 = Q_4 = \emptyset$	end of first iteration
				SAME		after second call to REARRANGE
(8,7,6,5,4,3)	(6)	(3,4,1,2)	(4)	(2,4,0,0)	$Q_1 = [(1,0,2)]$ $Q_2 = [(2,0,4)]$ $Q_3 = Q_4 = \emptyset$	end of second iteration
				SAME		after third call to REARRANGE
(8,7,6,5,4)	(5)	(4,1,2,3)	(4)	(2,4,4,0)	$Q_1 = [(1,0,2)]$ $Q_2 = [(2,0,4)]$ $Q_3 = [(3,0,4)]$ $Q_4 = \emptyset$	end of third iteration
(6,5,4)	(3)	(4,3)	(2)	(15,15,4,2)	$Q_1 = [(1,0,2), (8,2,15)]$ $Q_2 = [(2,0,4), (7,4,15)]$ $Q_3 = [(3,0,4)]$ $Q_4 = [(8,0,1), (7,1,2)]$	after the fourth call to REARRANGE
(6,5)	(2)	(3,4)	(2)	(15,15,4,8)	$Q_1 = [(1,0,2), (8,2,15)]$ $Q_2 = [(2,0,4), (7,4,15)]$ $Q_3 = [(3,0,4)]$ $Q_4 = [(8,0,1), (7,1,2), (4,2,8)]$	end of fourth iteration (figure 1a)

*The tuple (i, s, f) in Q_j indicates that τ_i is to be executed by P_j from time s to f.

(5)	(1)	(3)	(1)	(15,15,7,15)	Q_1 = [(1,0,2), (8,2,15)] Q_2 = [(2,0,4), (7,4,15)] Q_3 = [(3,0,4), (6,4,7)] Q_4 = [(8,0,1), (7,1,2), (4,2,8), (6,8,15)]	after the fifth call to REARRANGE
ϕ	(0)	(3)	(1)	(15,15,15,15)	Q_1 = [(1,0,2), (8,2,15)] Q_2 = [(2,0,4), (7,4,15)] Q_3 = [(3,0,4), (6,4,7), (5,7,15)] Q_4 = [(8,0,1), (7,1,2), (4,2,8), (6,8,15)]	end of the fifth iteration (see figure 1b)

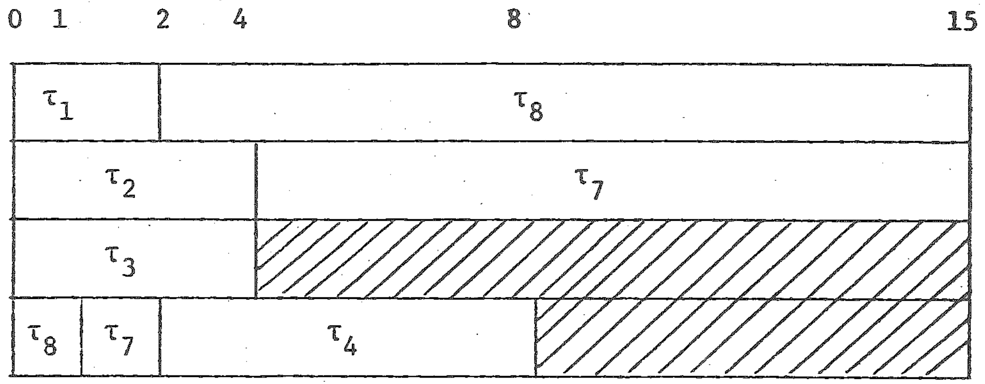

Figure 1a. Schedule including τ_1, τ_2, τ_3, τ_4, τ_7 and τ_8

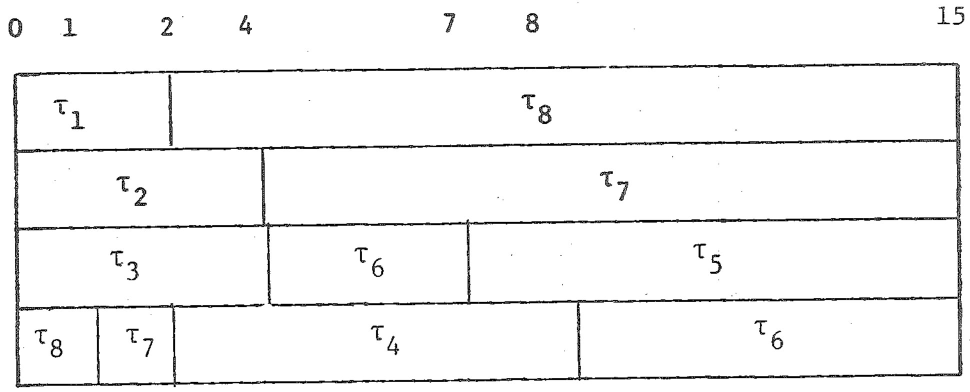

Figure 1b. Preemptive schedule for example 1 produced by algorithm Iβ:OMFT

Theorem 1: For every system of $m \geq 1$ identical processors, $n \geq m$ independent tasks and a deadline $\beta \geq \max\{t_n, \Sigma t_i/m\}$, algorithm $I\beta$:OMFT constructs a β:OMFT preemptive schedule.

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

Let S be the schedule constructed by algorithm $I\beta$:OMFT.* The last time line 5 was executed, q was zero and a1-a10 hold true (lemma 1). Clearly, S is a feasible schedule (a1) and $ft(S) \leq \beta$(a2). 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 5-11 was executed. The value for ℓ at the end of the kth iteration will be denoted by ℓ_k.

At the end of the kth iteration $(1 \leq k \leq r)$, we have that a1-a10 hold true (lemma 1). Clearly,

$$a_q = k + 1 \quad \text{(a6)} \tag{5}$$

$$\text{and} \quad f_{a_q-1+j-\ell_k} = \mu_{i_j} \quad \text{for} \quad 1 \leq j \leq \ell \quad \text{(a8-a9)}. \tag{6}$$

Note that $f_j = 0$ for $j \leq 0$. Substituting (5) in (6) and adding all equations in (6) we obtain (7)

$$\sum_{j=1}^{\ell} f_{k+j-\ell_k} = \sum_{j=1}^{\ell} \mu_{i_j} \tag{7}$$

Adding (7) together with $(m - \ell_k)\beta = \sum_{j \varepsilon M-\{i_1, i_2, \ldots, i_{\ell_k}\}} \mu_j$ (a2).

$$(m - \ell_k)\beta + \sum_{j=1}^{\ell_k} f_{k+j-\ell_k} = \sum_{j=1}^{m} \mu_j . \tag{8}$$

*Note that $I\beta$:OMFT terminates after at most n iterations.

Substituting $\sum_{j=1}^{m} \mu_j = \sum_{j \varepsilon N-\{a_1, \ldots, a_q\}} t_z$ (a4) in (8) and changing the order of summation,

$$(m - \ell_k)\beta + \sum_{j=1}^{\ell_k} f_{k-j+1} = \sum_{j \varepsilon N-\{a_1, a_2, \ldots, a_q\}} t_z \tag{9}$$

As $a_q = k + 1$ (5), it follows from a6 that $a_1 = k + q$. q was initially n. Every iteration (loop 5-11) it is decreased by 1. In procedure REARRANGE it is decreased by 1 each time ℓ is decreased by 1. ℓ was initially set to m. Hence $q = n - k - m + \ell_k$. Consequently $N - \{a_1, \ldots, a_q\} = \{1, 2, \ldots, k\} \cup \{n - m + \ell_k + 1, \ldots, n\}$. Substituting in (9) we obtain (10)

$$(m - \ell_k)\beta + \sum_{j=1}^{\ell_k} f_{k-j+1} = T_k + {}_{(n-m+\ell_k+1)}T \tag{10}$$

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

For $k = r$ we have that $0 = q = n - r - m + \ell_r$. Substituting in (10)

$$(m - \ell_r)\beta + \sum_{j=1}^{\ell_r} f_{r-j+1} = T_n$$

Using this equation for $r + 1 \leq k \leq n$ we obtain

$$(m - \ell_r - (k - r))\beta + \sum_{j=1}^{\ell_r+(k-r)} f_{k-j+1} \leq T_n \tag{11}$$

as $f_{k+1} \leq \beta$, $f_{k+2} \leq \beta$, $\ldots$, $f_n \leq \beta$.

The inequalities generated by example 1 are shown below

$r = 5$

$k = 1, \ell_1 = 4$	$f_1 = T_1$
$k = 2, \ell_2 = 4$	$f_1 + f_2 = T_2$
$k = 3, \ell_3 = 4$	$f_1 + f_2 + f_3 = T_3$
$k = 4, \ell_4 = 2$	$2\beta + f_3 + f_4 = T_4 + (7)^T$
$k = 5, \ell_r = 1$	$3\beta + f_5 = T_5 + (6)^T$
$k = 6$	$2\beta + f_5 + f_6 \le T_8$
$k = 7$	$\beta + f_5 + f_6 + f_7 \le T_8$
$k = 8$	$f_5 + f_6 + f_7 + f_8 \le T_8$

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

From lemma 3 it follows that $f_1' \le f_2' \le \dots \le f_n'$, where f_i' is the completion time for task τ_i in schedule S'. Now, we will obtain inequalities for schedule S'. Lemma 4 will be used r times. For $1 \le i \le r$, the value of k to be used in lemma 4 is $m - \ell_i + 1$ and w is i. From 2) in lemma 4 we obtain

$$(m - \ell_i)\beta + \sum_{j=i-\ell_i+1}^{i} f_j' \ge T_i + (n-m+\ell_i+1)^T$$

$$(m - \ell_i)\beta + \sum_{j=1}^{\ell_i} f_{i-j+1}' \ge T_i + (n-m+\ell_i+1)^T. \qquad (12)$$

Inequalities for $r + 1 \le i \le n$, are obtained using part 1) of lemma 4 with $k = m - \ell_r + 1$ and $w = r$.

$$f_n' + f_{n-1}' + \dots + f_{n-m+\ell_r+1}' + f_r' + \dots + f_{r-\ell_r+1}' \ge T_r + (n-m+\ell_r+1)^T.$$

As $r + 1 = n - m + \ell_r + 1$ (using $0 = q = n - r - m + \ell_r$) we obtain

$$\sum_{j=i+1}^{n} f'_j + \sum_{j=1}^{\ell_r+(i-r)} f'_{i-j+1} \geq T_n .$$

As $ft(S') \leq \beta$, then $\beta \geq f'_n, \ldots, \beta \geq f'_{i+1}$. Substituting in the above equation,

$$(m - \ell_r - (i - r))\beta + \sum_{j=1}^{\ell_r+(i-r)} f'_{i-j+1} \geq T_n . \qquad (13)$$

Combining (10) and (11) together with (12) and (13) we obtain

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

where $a_{j,i} \leq a_{j+1,i}$ for $1 \leq j < i$ and $a_{i,i} > 0$. Using lemma 5 ($\delta = m$) it then follows that

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

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

Lemma 3 is stronger than theorem 3 given in [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.

<u>Lemma 3</u>: Any schedule S for an identical processor system can be transformed to a preemptive schedule S' with the following properties:

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

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

1) $f'_n + f'_{n-1} + \ldots + f'_{n-k+2} + f'_w + \ldots + f'_{w-m+k} \geq T_w + {}_{(n-k+2)}T$ and

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

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

Let R represent the shaded area of the schedule represented in figure 2. Clearly

$$f'_n + f'_{n-1} + \ldots + f'_{n-k+2} + f'_w + \ldots + f'_{w-m+k+1} + f'_{w-m+k}$$
$$= T_w + {}_{(n-k+2)}T + X + \sum_{i=w+1}^{n-k+1} \delta_i - \sum_{i=1}^{w} \rho_i - \sum_{i=n-k+2}^{n} \rho_i$$

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

ρ_i total time τ_i is scheduled outside R, for $i = 1, 2, \ldots, w, n-k+2, \ldots, n$.

and δ_i total time τ_i is scheduled inside R, for $w+1 \leq i \leq n-k+1$.

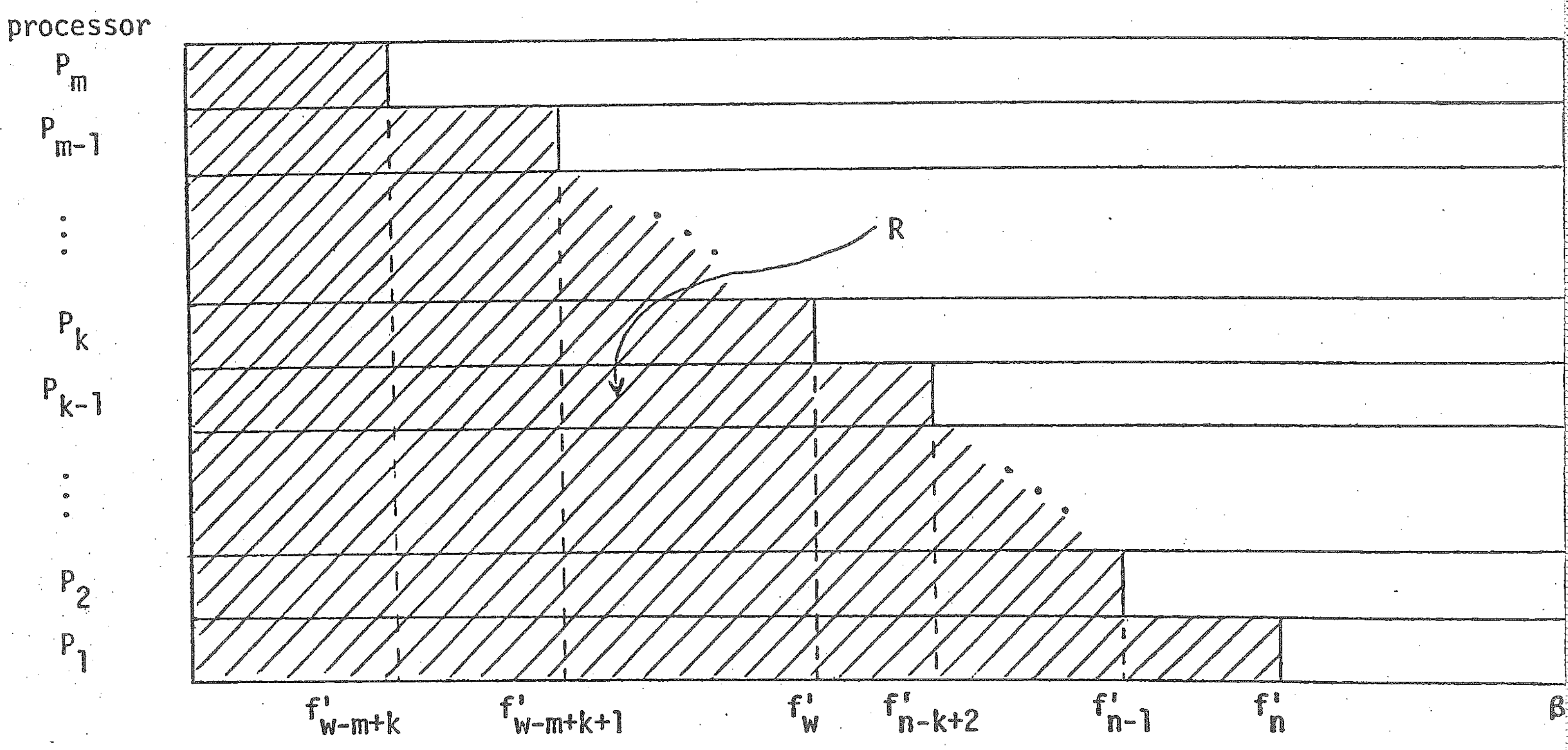

Figure 2: Schedule S'. In case $k = 1$, $f'_n, \ldots, f'_{n-k+2}$ are not included. R is the shaded region.

To prove 1) it is required to show that

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

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

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

ii) If ℓ of the machines are inside R, then only ℓ tasks of $\tau_{w-m+k}, \ldots, \tau_w, \tau_{n-k+2}, \ldots, \tau_n$ have not yet been completed. Clearly these ℓ tasks are the only tasks that can contribute to ρ in this interval. If k_1 of these tasks execute outside R or do not execute at all, then k_1 of the machines inside R will be idle or execute $\tau_{w+1}, \ldots, \tau_{n-k+1}$. Hence a) holds for the interval.

So, 1) holds for any schedule S'.

To prove 2), we use 1) and the restriction $ft(S') \leq \beta$. This completes the proof of the lemma. □

Lemma 5: Given that

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

where $a_{j,i} \leq a_{j+1,i}$ for $1 \leq j < i$, $\delta > 0$, $a_{j,i} \geq 0$, $a_{i,i} > 0$, $f_i \geq 0$ 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 $i = n - 1, \ldots, 1$

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

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 Iβ:OMFT is $O(nm)$.

Proof: Each time procedure REARRANGE is invoked it takes time $O(k_in + m)$, where k_i is the number of times the main loop is executed. ℓ is initially m. Each time the main loop (REARRANGE) is executed, ℓ is decreased by 1. Once ℓ is one, the main loop is never repeated (see line 1). Hence $\Sigma k_i < m$. Consequently the overall time complexity for REARRANGE is $O(nm + km)$, where k is the number of calls. Lines 1-4 in Iβ:OMFT take time $O(n + m)$. Loop 5-11 is executed at most n times. Lines 8-11 take time $O(m)$. The total time taken by line 7 is $O(nm)$. Hence the overall time complexity for Iβ:OMFT is $O(nm)$. □

Theorem 3: The maximum number of preemptions introduced by algorithm Iβ:OMFT is $m - 1$.

Proof: Clearly, the only place where preemptions are introduced is in procedure REARRANGE. The main loop in REARRANGE is executed at most $m - 1$ times (see theorem 2). Each time no more than 1 preemption is introduced. Hence no more than $m - 1$ preemptions are introduced. $\square$

III. CONCLUSIONS

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, the algorithm reduces to the well known SPT rule. It can be easily shown that algorithm Iβ:OMFT obtains β:OWMFT preemptive schedules when the weights are agreeable. The proofs are similar to the ones in this paper. A similar algorithm can be adapted to obtain β:OMFT preemptive schedules for uniform machines [G2]. However, the algorithm is textually more complex and introduces $0(nm)$ preemptions. 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 scheduling," TR-214 Penn State University, Dec. 1976.

[G2] T. Gonzalez, "Minimizing the Mean and Maximum Finishing Time on Uniform Processors" (in preparation).

[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 and 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.