# CHAPTER VI

# UNIFORM PROCESSOR SYSTEMS

## 4.1 Introduction

A uniform processor system [15] is one in which the processors $P_1, P_2, \ldots, P_m$ have relative speeds $s_1, s_2, \ldots, s_m$ respectively. It is assumed that the speeds have been normalized such that $s_1 = 1$ and $s_i \geq 1$ for $2 \leq i \leq m$. The problem of scheduling n independent tasks $(J_1, J_2, \ldots, J_n)$ with execution times $(t_1, t_2, \ldots, t_n)$ on m uniform processors to obtain a schedule with the optimal (least) finish time is known to be NP-Complete [3,15]. Hence, it appears unlikely that there is any polynomial time bounded algorithm to generate such schedules. For preemptive scheduling, however, optimal finish time algorithms can be obtained in polynomial time [16, 30]. Horowitz and Sahni [15] showed that for any m, polynomial time algorithms exist to obtain schedules with a finish time arbitrarily close to the optimal finish time. The complexity of these algorithms was, however, exponential in m. The purpose of this Chapter is to study the finish time properties of LPT schedules with respect to the optimal finish time.

Definition 4.1.1

An LPT (Largest Processing Time) schedule is a

schedule obtained by assigning tasks to processors in order of nonincreasing processing times. When a task is being considered for assignment to a processor, it is assigned to that processor on which its finishing time will be earliest. Ties are broken by assigning the task to the processor with least index. ■

One may easily verify that for identical processor systems, this definition is equivalent to that of [4], p. 100. Graham [13] studied LPT schedules for the special case of identical processors, i.e., $s_i = 1$, $1 \leq i \leq m$. If $\hat{f}$ is the finish time of the LPT schedule and f* the optimal finish time, then Graham's result is that $\hat{f}/f^* \leq \frac{4}{3} - \frac{1}{3m}$ and that this bound is the best possible bound. In section 4.2 we extend his work to the general case of uniform processors. While the bound we obtain is best possible for m = 2, it appears that it is not so for m > 2. In view of this, we turn our attention to another special case of uniform processors i.e., $s_i = 1$, $1 \leq i < m$ and $s_m = s \geq 1$. This case has previously been studied by J.W.S. Liu and C.L. Liu [29]. Using a priority assignment according to lengths of tasks, they show that $f/f^* \leq \frac{2(m-1+s)}{s+2}$ for $s \leq 2$ and $f/f^* \leq \frac{m-1+s}{2}$ for $s \geq 2$, where f is the finish time of the priority schedule.

Similar bounds for list schedules are also obtain

ed by them. We are able to show that for $m \geq 3$ $\hat{f}/f^* \leq 3/2 - 1/(2m)$ and that this bound is the best possible for $m = 3$. For $m > 3$ we conjecture that $\hat{f}/f^* \leq 4/3$.

Before presenting our results we develop the necessary notation and basic results. If S is the set of tasks being scheduled, then it will sometimes be necessary to distinguish between finish times of different sets of tasks. To do this S will appear as a superscript along with $\hat{f}$ or $f^*$ as in $\hat{f}^S$ and $f^{*S}$. If the number of processors is important, then this number will appear as a subscript as in $\hat{f}_m$, $f_m^{*S}$ etc. We shall refer to the sets of tasks (jobs) by their task execution time. Thus we speak of a set, S, of tasks $(t_1 \geq t_2 \geq \ldots \geq t_n)$ meaning the execution time of task $J_i$ is $t_i$ and $t_i \geq t_{i+1}$, $1 \leq i < n$. The m processors $P_1, P_2, \ldots, P_m$ are assumed ordered such that $s_1 = 1$ and $1 \leq s_i \leq s_{i+1}$, $2 \leq i < m$. The following result from [4, p. 102] is made use of:

Lemma 4.1.1 If for any m $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ is the smallest set of tasks for which $\hat{f}/f^* > k$ then $t_n$ determines the finish time $\hat{f}$ (i.e. task n has the latest completition time).

Proof Appears in [4] p. 102. ■

## 4.2 Basic Results

In this section, we prove two important lemmas that are used throughout the Chapter (Lemmas 4.2.2 and 4.2.3). We also derive the bound $2m/(m+1)$ for the ratio $\hat{f}/f^*$ for the general m-processor system. Examples are shown for which $\hat{f}/f^*$ approaches 3/2 as $m \to \infty$.

We begin with the following lemma, informally, it states that if either the LPT or optimal schedule of an (m+1)-processor system has an idle processor, then the ratio $\hat{f}/f^*$ for this schedule is no worse that $\hat{f}/f^*$ For m processors.

Lemma 4.2.1 For $m \geq 1$, let $g(m, s_2,\ldots,s_m)$ be such that $\hat{f}_m/f^*_m \leq g(m, s_2,\ldots,s_m)$. Consider any (m+1)-processor system with job set $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ and processor speeds $1 = s_1 \leq s_2 \leq \ldots \leq s_{m+1}$. If a processor is idle in either the LPT or optimal schedule of S, then $\hat{f}^S_{m+1}/f^{*S}_{m+1} \leq g(m, s_3/s_2,\ldots, s_{m+1}/s_2)$.

Proof Suppose in the LPT schedule of S a processor $P_i$ is idle. Then it must be the case that in the optimal schedule, $P_i$ is also idle. Otherwise, $\hat{f}^S_{m+1} \leq t_n/s_i$, $f^{*S}_{m+1} \geq t_n/s_i$ and $\hat{f}^S_{m+1}/f^{*S}_{m+1} = 1$. So we need only consider the case when $P_i$ is idle in the optimal schedule. If $P_i$ is idle then clearly $P_1$ is also idle or can be made idle without increasing $f^*$ by scheduling the jobs on $P_1$ onto $P_i$. Consider the

m-processor system with job set $S$ and processor speeds $1 = s_2/s_2 \leq s_3/s_2 \leq \ldots \leq s_{m+1}/s_2$. Then by assumption, for this system, $\hat{f}_m^S/f_m^{*S} \leq g(m, s_3/s_2, \ldots, s_{m+1}/s_2)$. Moreover, $\hat{f}_{m+1}^S \leq s_2 \hat{f}_m^S$ and $f_{m+1}^{*S} = s_2 f_m^{*S}$. It follows that $\hat{f}_{m+1}^S/f_{m+1}^{*S} \leq g(m, s_3/s_2, \ldots, s_{m+1}/s_2)$. ■

The next lemma gives an estimate of $\hat{f}/f^*$ for the case when $\hat{f}$ is determined by the job with the smallest execution time.

Lemma 4.2.2 Consider an m-processor system with job set $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ and speeds $s_1, s_2, \ldots, s_m$. If in the LPT schedule of S, the finish time $\hat{f}$ is determined by $t_n$, (i.e., if task n has the latest completition time) then $\hat{f}/f^* \leq 1 + \frac{(m-1)t_n}{Qf^*}$, where $Q = \Sigma s_i$.

Proof Let the LPT schedule be as shown in Fig. 4.2.1, where $P_k$ determines the finish time. Each $T_i$ is the sum (possibly 0) of execution times of jobs scheduled on $P_i$ prior to $t_n$'s assignment, $T_1 + \ldots + T_m = t_1 + \ldots t_{n-1}$.

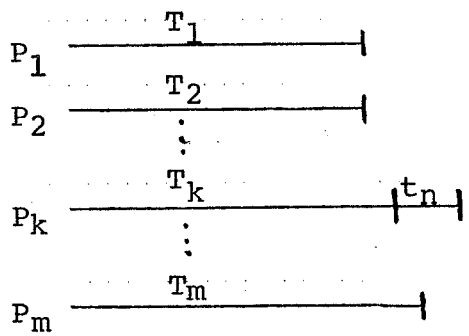



Fig. 4.2.1

Since task n determines the finish time, $\hat{f} = (T_k + t_n)/s$ and $\frac{T_i+t_n}{s_i} \geq \hat{f}$ for $i \neq k$. Hence, $\hat{f}s_i - T_i \leq t_n$ and so $\hat{f} \sum_{i\neq k} s_i - \sum_{i\neq k} T_i \leq (m-1)t_n$. This, together with $\hat{f}s_k = T_k + t_n$ yields

$$\begin{aligned} \hat{f}Q &\leq T_i + mt_n \\ &= t_i + (m-1)t_n. \end{aligned}$$

Since $f^* \geq t_i/Q$, we get $\hat{f}/f^* \leq 1 + \frac{(m-1)t_n}{f^*Q}$ ■

Using Lemmas 4.2.1 and 4.2.2, we can now derive a bound for the m processor system.

Theorem 4.2.1 For an m-processor system, $\hat{f}/f^* \leq \frac{2m}{m-1}$.

Proof For $m = 1$, the theorem obviously holds. Now suppose the theorem holds for $1,2,\ldots,m-1$ processors but fails for m-processors. Let $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ be the smallest set of jobs which gives a bound $\hat{f}_m/f^*_m > \frac{2m}{m+1}$. Then by Lemma 4.1.1, $t_n$ determines the finish time. There are two cases to consider. Both lead to a contradiction.

Case 1 $n \geq m + 1$. Then by Lemma 4.2.2,

$$\begin{aligned} f_m/f^*_m &\leq 1 + \frac{(m-1)t_n}{Qf^*} \\ &\leq 1 + \frac{(m-1)t_n}{Q(\frac{\Sigma t_i}{Q})} \end{aligned}$$

$$\leq 1 + \frac{(m-1)t_n}{nt_n} = 1 + \frac{(m-1)}{n} \leq 1 + \frac{m-1}{m+1} = \frac{2m}{m+1} \quad ,$$

a contradiction.

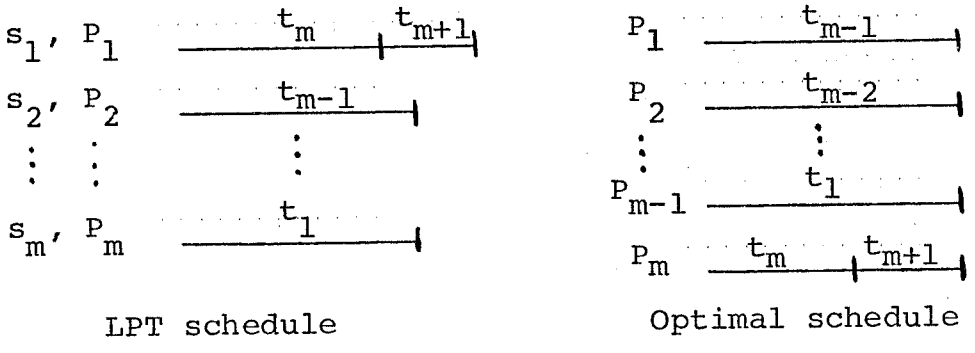
Case 2 $n \leq m$ . Then in the optimal schedule, either each processor has exactly one job or a processor is idle. In the first case, $\hat{f}_m / f^*_m = 1$, since no processor can be idle in the LPT schedule (see proof of Lemma 4.2.1). For the second case $\hat{f}^S_m / f^{*S}_m \leq \hat{f}^S_{m-1} / f^{*S}_{m-1}$ $\leq \frac{2(m-1)}{m} \leq \frac{2m}{m+1}$ by Lemma 4.2.1. Either case leads to a contadiction. ■

Corollary 4.2.1 For an m-processor system, $\hat{f}/f^* < 2$ .

The bound of Theorem 4.2.1 is probably not a tight bound. However, we can show that there are examples approaching the bound 1.5 as $m \to \infty$ . ■

Theorem 2.2 For every $m \geq 2$, there is an example of an m-processor system and a set of jobs $S$ for which $\hat{f}^S / f^{*S} = c$, where $c$ is a positive root of the equation $2s^m - s^{m-1} - \ldots - s - 2 = 0$ .

Proof The example we shall construct has job set $S = (t_1 \geq t_2 \geq \ldots \geq t_m \geq t_{m+1})$ (where $m$ is the number of processors) and processor speeds $1 = s_1 \leq \ldots \leq s_m$. The $t_i$'s and $s_i$'s will satisfy the following properties ( see Fig. 4.2.2 ):




LPT schedule Optimal schedule

Fig. 4.2.2

(4.2.1) $\hat{f} = t_m + t_{m+1}$ and $f^* = \frac{t_m + t_{m+1}}{s_m}$

(4.2.2) $t_m = t_{m+1} = t$ ,

(4.2.3) $t_m + t_{m+1} = 2t = \frac{t_i + t}{s_{m-i+1}}$ for $1 \leq i \leq m - 1$ ,

(4.2.4) $\frac{t_m + t_{m+1}}{s_m} = \frac{2t}{s_m} = \frac{t_i}{s_{m-i}}$ for $1 \leq i \leq m - 1$ .

Then $\hat{f}/f^* = \frac{2t}{\frac{2t}{s_m}} = s_m$ . From properties (4.2.1) - (4.2.4) we can derive the equation for $s_m$ . From (4.2.3) we get:

(4.2.5) $t_i = 2ts_{m-i+1} - t = t(2s_{m-i+1} - 1)$ .

(From 4.2.4) we have

(4.2.6) $s_m t_i = 2ts_{m-i}$

(4.2.5) and (4.2.6) yields

(4.2.7) $s_{m-i+1} = \frac{2s_{m-i} + s_m}{2s_m}$ for $1 \leq i \leq m - 1$ .

Using (4.2.7) repeatedly for $i=1,2,\ldots,m-1$ we get:

$$s_m = \frac{2s_{m-1} + s_m}{2s_m}$$

$$= \frac{2\left(\frac{2s_{m-2} + s_m}{2s_m}\right) + s_m}{2s_m}$$

$$= \frac{2s_{m-2} + s_m + s_m^2}{2s_m^2}$$

$$= \frac{2\left(\frac{2s_{m-3} + s_m}{2s_m}\right) + s_m + s_m^2}{2s_m^2}$$

$$= \frac{2s_{m-3} + s_m + s_m^2 + s_m^3}{2s_m^3}$$

$$\vdots$$

$$= \frac{2s_1 + s_m + s_m^2 + \ldots + s_m^{m-1}}{2s_m^{m-1}}$$

Hence

$$s_m = \frac{2 + s_m + s_m^2 + \ldots + s_m^{m-1}}{2s_m^{m-1}} \qquad \text{(since } s_1 = 1\text{)}$$

or

(4.2.8) $\quad 2s_m^m - s_m^{m-1} - s_m^{m-2} - \ldots - s_m - 2 = 0 .$

The polynomial on the left hand side of (4.2.8) has one sign change and so from Descartes rule it also has one positive real root. The root must clearly be $> 1$ as otherwise the right hand side is $< 0$.

Let $c$ be a positive root of equation (4.2.8). We can construct an example of an m-processor system with $\hat{f}/f^* = c$ by setting $s_m = c$ and computing $s_2, \ldots, s_{m-1}$ in terms of $c$ using (4.2.7). (Of course, $s_1 = 1$.) Then by letting $t_m = t_{m+1} = t$, we can determine the

values of $t_1, t_2, \ldots, t_{m-1}$ in terms of t using (4.2.4). ■

Corollary 4.2.2 There exist uniform processor systems and job sets S for which $\hat{f}/f^* \simeq 1.5$ .

Proof From Theorem 4.2.1 we know that there are jobs sets, S , for which $\hat{f}/f^* = c$ where c is a positive root of (4.2.8). Let s be a root. Rearranging terms, we get:

$$2s^m - 1 = \sum_{0 \le i < m} s^i$$

$$= \frac{s^m - 1}{s-1}$$

or $2s^{m+1} - 3s^m - s + 2 = 0$ .

Since $s > 1$ , for $m \to \infty$ we have $s \to 3/2$ as a root. ■

Example 4.2.1

(a) $m = 2$ :

Then we have $2s_2^2 - s_2 - 2 = 0$ , where we find $s_2 = \frac{1+\sqrt{17}}{4}$ . Of course, $s_1 = 1$. Let $t_2 = t_3 = 1$. From equation (4.2.4), we find $t_1 = \frac{2t}{s_2} \cdot s_1 = \frac{8}{1+\sqrt{17}}$ . One easily verifies that $\hat{f}/f^* = \frac{1+\sqrt{17}}{4}$ .

(b) $m = 3$ :

The equation to use is $2s_3^3 - s_3^2 - s_3 - 2 = 0$. $s_3 = 1.384$ is an approximate root of this equation.

Using equation (4.2.7), we find $s_2 = \frac{s_3(2s_3-1)}{2} = 1.223$ and $s_1 = 1$. Let $t_3 = t_4 = t = 1$. Using equation (4.2.4), find $t_2 = \frac{2t}{s_3} \cdot s_2 = 1.767$ and $t_1 = \frac{2t}{s_3} \cdot s_1 = 1.445$. Again we can check that $f/f^*$ is approximately 1.384.

(c) Some other roots of (4.2.8) are 1.493 for $m = 10$ and 1.499 for $m = 20$. ■

## 4.3 Special Case (1, 1,...,1 , s)

In this section we study the special case in which all but one of the $m \geq 1$ processors has a speed of 1. The $m^{\text{th}}$ processor $P_m$ has a speed $s \geq 1$. The main result of this section is stated below as Theorem 4.3.1.

Theorem 4.3.1 For $m \geq 2$ the ratio $\hat{f}/f^*$ has the following bounds:

(i) $\hat{f}/f^* \leq (1 + \sqrt{17})/4$ for $m = 2$

(ii) $\hat{f}/f^* \leq 3/2 - 1/(2m)$ for $m > 2$ .

Proof (i) is proved in Lemma 4.3.2. (ii) follows from Lemmas 4.3.1 - 4.3.6 and the fact that the bound is a monotone increasing function in $m$. ■

Before proving the theorem we derive a general bound for $\hat{f}/f^*$ in terms of $m$ and $s$ .

Lemma 4.3.1 For an m-processor system with $s_i = 1$ for $1 \leq i < m$ and $s_m = s$ , $\hat{f}/f^* \leq \frac{2(m-1+s)}{m-1+2s}$ .

Proof If $m = 1$, the lemma is obviously true since $\hat{f}/f^* = 1$. Now assume that the lemma holds for $1,2,\ldots,$ $m-1$ processors but fail for $m$ $(m \geq 2)$. For this $m$, Let $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ be the smallest set of jobs for which $\hat{f}/f^* > \frac{2(m-1+s)}{m-1+2s}$. Suppose a processor is idle in either the LPT or optimal schedule of $S$.

Then $\hat{f}^S_m/f^{*S}_m \leq \hat{f}^S_{m-1}/f^{*S}_{m-1} \leq \frac{2(m-2+s)}{m-2+2s} < \frac{2(m-1+s)}{m-1+2s}$ by Lemma 4.2.1 .

So we may assume that no processor is idle in either the LPT or optimal schedule of $S$. We consider two cases, both leading to a contradiction.

Case 1 The LPT schedule is as shown in Fig. 4.3.1 , where each $T_i$ represents the sum of execution times of jobs scheduled on $P_i$ prior to the assignment of $t_n$, $T_1 + T_2 + \ldots + T_m = t_1 + t_2 + \ldots + t_{n-1}$. By assumption, no processor is idle. Hence $T_i > 0$ for $2 \leq i \leq m$. Since the first $m - 1$ processors have speed 1, we may assume that $T_i \geq T_1$ and $1 < i \leq m-1$. Now if $T_1 = 0$, then $\hat{f} = t_n$. But $f^* \geq t_n$ since by assumption no processor is idle in the optimal schedule. Then $\hat{f}/f^* = 1$. So we may also assume that $T_1 \geq t_n$. Then

$$\hat{f}/f^* \leq \frac{T_1 + t_n}{(\Sigma T_i + t_n)/(m-1+s)} \leq \frac{(m-1+s)(T_1 + t_n)}{(m-1)T_1 + T_m + t_n}$$

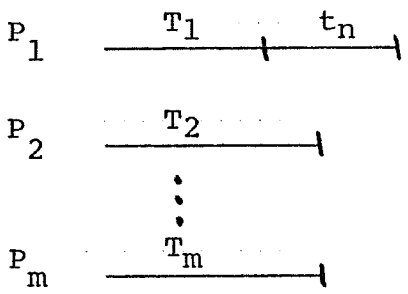



Fig. 4.3.1

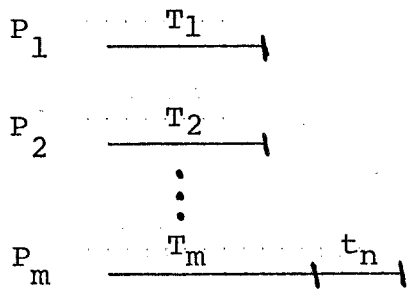



Fig. 4.3.2

Now, since $t_n$ determines the finish time,

$\frac{T_m + t_n}{s} \geq T_1 + t_n$ or $T_m \geq sT_1 + (s-1)t_n$ . Then

$$\hat{f}/f^* \leq \frac{(m-1+s)(T_1+t_n)}{(m-1)T_1+sT_1+(s-1)t_n+t_n}$$

$$= \frac{(m-1+s)(T_1+t_n)}{(m-1)T_1 + s(T_1+t_n)}$$

$$= \frac{m-1+s}{s + \frac{(m-1)T_1}{T_1+t_n}}$$

$$\leq \frac{m-1+s}{s + \frac{m-1}{2}}$$ since the minimum $\frac{T_1}{T_1+t_n}$ with the constrain $T_1 \geq t_n$ occurs when $T_1 = t_n$ .

Hence, $\hat{f}/f^* \leq \frac{2(m-1+s)}{m-1+2s}$ , a contradiction.

Case 2 Suppose the LPT schedule is as shown in Fig. 4.3.2, where we again assume that $T_i \geq T_1 \geq t_n$ . We may also assume that $T_m > 0$ ; otherwise $\hat{f}/f^* = 1$ since $\hat{f} = t_n/s$ .

Then $\hat{f}/f^* \leq \dfrac{\frac{T_m + t_n}{s}}{(\Sigma T_i + t_n)/(m-1+s)}$

$$\leq \frac{(m-1+s)\left(\frac{T_m + t_n}{s}\right)}{(m-1)T_1 + T_m + t_n}$$

Since $t_n$ is scheduled on $P_m$, $T_1 + t_n \geq \dfrac{T_m + t_n}{s}$.

We have two subcase:

(a) We can find a $T$ such that $t_n \leq T \leq T_1$ and $T + t_n = \dfrac{T_m + t_n}{s}$. Then

$$\hat{f}/f^* \leq \frac{(m-1+s)(T+t_n)}{(m-1)T + T_m + t_n}$$

$$= \frac{(m-1+s)(T+t_n)}{(m-1)T + s(T_1+t_n)}$$

$$= \frac{(m-1+s)}{s + \frac{(m-1)T}{T + t_n}}$$

$$\leq \frac{(m-1+s)}{s + \frac{m-1}{s}} = \frac{2(m-1+s)}{m-1+2s}$$

Again, we get a contradiction.

(b) If (a) is not possible, we let $T = t_n$. Then $T + t_n = 2t_n > \dfrac{T_m + t_n}{s}$ or $T_m < (2s-1)t_n$. Then

$$\hat{f}/f^* \leq \frac{(m-1+s)\left(\frac{T_m + t_n}{s}\right)}{(m-1)T_1 + T_m + t_n}$$

$$\leq \left(\frac{m-1+s}{s}\right) \frac{T_m + t_n}{(m-1)t_n + T_m + t_n}$$

$$= \left(\frac{m-1+s}{s}\right) \frac{1}{1 + \frac{(m-1)t_n}{T_m + t_n}}$$

$$\leq \left(\frac{m-1+s}{s}\right) \frac{1}{1 + \frac{(m-1)t_n}{(2s-1)t_n + t_n}}$$

$$= \frac{2(m-1+s)}{m-1+2s} \quad , \quad \text{a contradiction.} \blacksquare$$

The bound for $m = 2$ follows from the following lemma.

Lemma 4.3.2 For an $m$ processor system with $s_i = 1$, $1 \leq i < m$ and $s_m = s$, $\hat{f}/f^* \leq \frac{(3-m) + \sqrt{(3-m)^2 + 16(m-1)}}{4}$ Moreover, for $m = 2$, the bound is tight.

Proof Let $k > 1$ be the desired bound for $\hat{f}/f^*$. Let $Q = \Sigma s_i = m-1+s$. First we show that if $s \leq \frac{2Q(k-1)}{m-1}$, then $\hat{f}/f^* \leq k$. Suppose not. Let $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ be the smallest set of jobs for which $\hat{f}/f^* > k$. Then $t_n$ determines the finish time and by Lemma 4.2.2, $\hat{f}/f^* \leq 1 + \frac{(m-1)t_n}{Qf^*}$. Hence $f^* < \frac{(m-1)t_n}{Q(k-1)}$. It follows that the number of jobs on each processor in the optimal schedule of $S$ is less than $\frac{(m-1)s}{Q(k-1)} \leq 2$. But then this case, $\hat{f}/f^* = 1$. This contradicts the assumption that $S$ produces a bound $> k$. Thus if $s \leq \frac{2Q(k-1)}{m-1}$, then $\hat{f}/f^* \leq k$. This, in turn, implies that if $Q \leq (m-1) + \frac{2Q(k-1)}{m-1}$, then $\hat{f}/f^* \leq k$ or that

(4.3.1) if $Q \leq \frac{(m-1)^2}{m-2k+1}$ then $\hat{f}/f^* \leq k$. Now by Lemma 4.3.1, we have $\hat{f}/f^* \leq \frac{2(m-1+s)}{m-1+2s} = \frac{2(m-1+s)}{2(m-1+s)-(m-1)}$

$= \frac{2Q}{2Q-(m-1)}$ . It follows that if $\frac{2Q}{2Q-(m-1)} \leq k$ , then $\hat{f}/f^* \leq k$ or

(4.3.2) if $Q \geq \frac{(m-1)k}{2(k-1)}$ then $\hat{f}/f^* \leq k$ .

To satisfy (4.3.1) and (4.3.2) simultaneously, we must have $\frac{(m-1)^2}{m-2k+1} = \frac{(m-1)k}{2(k-1)}$ , from which we get $k = \frac{(3-m) + \sqrt{(3-m)^2 + 16(m-1)}}{4}$ . In this case $\hat{f}/f^* \leq k$ for all $Q$ .

For the case $m = 2$ , we have $k = \frac{1 + \sqrt{17}}{4}$ which is tight since we have seen an example for which the bound is achieved. ■

In arriving at the proof of the theorem for $m > 2$, it is necessary to prove four lemmas. To begin with, we show that if for any set of jobs, $S$ , an optimal schedule has more than one job on any of the processors $P_1, P_2, \ldots, P_{m-1}$ then $\hat{f}^S/f^{*S} \leq 3/2 - 1/(2m)$.

Lemma 4.3.3 For any set of jobs, $S$ , either

(i) processors $P_1$-$P_{m-1}$ have at most one job scheduled on each in every optimal schedule

or (ii) $\hat{f}_m^S/f_m^{*S} \leq 3/2 - 1/(2m)$ .

Proof Suppose (ii) is not true for some set of jobs. Let $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ be the smallest set of jobs for which $\hat{f}_m^S/f_m^{*S} > 3/2 - 1/(2m)$ . From Lemma 4.2.2 we get

$$\hat{f}_m^S / f_m^{*S} \leq 1 + \frac{(m-1)\,t_n}{(m-1+s)\,f_m^*} > 3/2 - 1/(2m)$$

or

$$\frac{(m-1)\,t_n}{(m-1+s)\,f_m^*} > \frac{m-1}{2m}$$

or

$$t_n > \frac{m-1+s}{2m}\, f_m^*$$

$$\geq (1/2)\, f_m^*$$

I.e., $f_m^* < 2t_n$ which, in turn, means that none of the processors $P_1$-$P_{m-1}$ can have more than one job scheduled on them in an optimal schedule. ■

Next, we prove that if $s \geq m-1$ then $\hat{f}/f^* \leq 4/3$.

Lemma 4.3.4 If $s \geq m-1$ then $\hat{f}/f^* \leq 4/3 \leq \frac{3}{2} - \frac{1}{2m}$ for $m > 2$ .

Proof Lemma 4.3.1 gives

$$\hat{f}/f^* \leq \frac{2(m-1+s)}{m-1+2s}$$

The right hand side of the above inequality is a decreasing function of $s$ . Hence, for $s \geq m-1$ we obtain

$$\hat{f}_m / f_m^* \leq \frac{4m-4}{3(m-1)}$$

$$= 4/3$$

$$\leq 3/2 - 1/(2m) \qquad m > 2 \text{ .}$$ ■

As a result of Lemmas 4.3.3 and 4.3.4 the only counter examples to Theorem 4.3.1 are sets of jobs, $S$, for which the optimal schedules have at most one job on each of $P_1 - P_{m-1}$ and the speed, $s$ , of $P_m$ is <m-1.

The next two lemmas show that for this kind of an optimal and $s < m-1$ the bound of theorem 4.3.1 cannot be violated.

Lemma 4.3.5 Let $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ be the smallest set of jobs for which $\hat{f}/f^* > 3/2 - 1/(2m)$. If in the LPT schedule, $t_i$ is the only job scheduled on one of the processors, $P_1, P_2, \ldots, P_{m-1}$ and if in an optimal schedule $t_j$ is the only job scheduled on one of the processors, $P_1, P_2, \ldots, P_{m-1}$ then, either

$$\text{(i)} \quad \hat{f}_m^S / f_m^{*S} \leq f_{m-1}/f_{m-1}^*$$

or

$$\text{(ii)} \quad t_i < t_j$$

Proof From Lemma 4.1.1 it follows that $t_n$ determines the finish time $\hat{f}^S$. If anyone of the processors $P_1, P_2, \ldots, P_m$ is idle in an optimal solution (i.e. no jobs have been scheduled on it) then $f_m^* = f_{m-1}^*$. But, $\hat{f}_m^S \leq \hat{f}_{m-1}^S$ and so $\hat{f}_m^S / f_m^{*S} \leq \hat{f}_{m-1}^S / f_{m-1}^{*S}$. We may therefore assume that no processor is idle in any optimal solution. Hence, $f_m^{*S} \geq t_n$. If $i = n$ then $\hat{f}_m^S = t_n$ (as $t_i$ is the only job on some processor $P_1, P_2, \ldots, P_{m-1}$) and $\hat{f}_m^S / f_m^{*S} \leq 1$. Therefore $i \neq n$. Now, we have

$$f_m^{*S} = \max\{t_j, f_{m-1}^{*S-\{t_j\}}\}$$

$$\geq f_{m-1}^{*S-\{t_j\}}$$

$$\geq f^{*S-\{t_i\}}_{m-1} \quad \ldots \text{ as } t_i \geq t_j$$

but, $$\hat{f}^{S}_{m} = \hat{f}^{S-\{t_i\}}_{m-1} \quad \ldots \text{ as } i \neq n$$

$$\therefore \quad \hat{f}^{S}_{m}/f^{*S}_{m} \leq \hat{f}^{S-\{t_i\}}_{m-1}/f^{*S-\{t_i\}}_{m-1}$$

$$\leq \hat{f}_{m-1}/f^{*}_{m-1} \quad \blacksquare$$

Lemma 4.3.6 When $s < m-1$ and an optimal schedule for any set of jobs $S$ has at most one job on each of processors $P_1$-$P_{m-1}$ then $\hat{f}_m/f^*_m \leq 3/2 - 1/(2m)$ .

Proof Let $S = (t_1 \geq t_2 \geq \ldots \geq t_n)$ be the smallest set of jobs and $m$ the least $m > 2$ for which the lemma is not true. From Lemma 4.3.1 we obtain

$\hat{f}/f^* \leq 1 + \frac{m-1}{m-1+s} \frac{t_n}{f^*}$. By assumption $\hat{f}/f^* > 3/2 - 1/(2m)$. Therefore,

$$1 + \frac{(m-1)}{m-1+s} \frac{t_n}{f^*} > 3/2 - 1/(2m)$$

or

$$f^* < \frac{2m}{m-1+s} t_n \quad \ldots \quad (4.3.3)$$

If $\#_m$ is the number of jobs on $P_m$ in an optimal schedule then, $f^* \geq \#_m t_n/s$ . Substituting this inequality into (4.3.3) yields:

$$\#_m < \frac{2sm}{m-1+s} \quad \ldots \quad (4.3.4)$$

The right hand side of the inequality (4.3.4) is an increasing function of $s$ . Since $s < m-1$ (4.3.4) yields the following bound on $\#_m$ :

$$\#_m < \frac{2(m-1)m}{2(m-1)} = m \ .$$

The optimal schedule has at most one job on each of $P_1$-$P_{m-1}$ . Hence, $n \leq 2m-2$.

The remainder of the proof shows that if $n \leq 2m-2$ then Lemma 4.3.5 can be used to show that $\hat{f}^S_m/f^{*S}_m \leq \hat{f}^S_{m-1}/f^{*S}_{m-1}$ thus contradicting the assumption that this was the least $m$ for which the lemma was false . (The contradiction comes about as $3/2 - 1/(2m)$ is monotone increasing in $m$ and the fact that when $m = 3$ this bound is $4/3$ which is greater than the known bound for $m = 2$ .) Clearly, we may assume that each processor has at least one job scheduled on it in every optimal schedule.

Let $k$ be the smallest index (i.e. largest job) on any of the processors $P_1$-$P_{m-1}$ in an optimal schedule. Then, the schedule obtained by assigning job $t_{k+i-1}$ to processor $P_i$ , $1 \leq i < m$ and the remaining jobs to processor $P_m$ has a finish time no greater than the optimal finish time $f^{*S}_m$ . Such a schedule shall be denoted by $OPT_k$ . Clearly, $1 \leq k \leq n-m+2$ . Since, $n \leq 2m - 2$ at least one of the processors $P_1$-$P_{m-1}$ has exactly one job scheduled on it (every processor must have at least one job on it as otherwise , by the definition of LPT $\hat{f} \leq t_n$ but $f^* \geq t_n$). Let the index of this job be $i$ . Then, $t_i$ must be the

largest job amongst jobs scheduled on $P_1$-$P_{m-1}$ in the LPT schedule (this again follows from the definition of LPT). But, $s < m-1$ implies $t_i \geq t_{m-1}$ as LPT cannot schedule all of the first $m-1$ jobs on $P_m$ when $s < m-1$. For all $k \geq 1$, $OPT_k$ has a job with index $j = k + m - 2 \geq m - 1$ on $P_{m-1}$ and this is the only job on $P_{m-1}$. By the ordering on the jobs, $t_j \leq t_{m-1}$. So, $t_i \geq t_j$. Lemma 4.3.5 now implies that $\hat{f}^S_m/f^{*S}_m \leq \hat{f}_{m-1}/f^*_{m-1}$; a contradiction. ■

Having shown that $\hat{f}/f^*$ is indeed bounded as in Theorem 4.3.1, the next question is: How good is the bound. From the previous section we know that the bound for $m = 2$ is tight. Lemma 4.3.7 shows that the bound is also tight for $m = 3$ and that for all $m > 3$ it is possible to have an $\hat{f}/f^*$ arbitrarily close to 4/3. Lemma 4.3.8 shows that for $m = 4$ and 5 there is no set of jobs $S$ for which $\hat{f}/f^* > 4/3$. This shows that the bound of $3/2 - 1/(2m)$ is not a tight bound for all values of $m$ and leads us to conjecture that for $m \geq 3$ the bound is in fact $4/3$. Note the closeness of this bound of $4/3$ to the bound $4/3-1/(3m)$ obtained by Graham [13] for the case of $s = 1$ (i.e. $m$ identical processors).

Lemma 4.3.7 For $m \geq 3$ and any $\varepsilon > 0$, there is a set of jobs, $S$, and a speed $s > 1$ for which

$\hat{f}/f^* > 4/3 - \varepsilon$ .

Proof For any $m \geq 3$ consider the set of jobs $t_1 = 1.5$ $t_2 = 1.5$ , $t_j = 1$ , $3 \leq j \leq m+2$ and $s = 2 + \varepsilon'$ with $\varepsilon'$ very close to zero . The LPT schedule has jobs $t_1$, $t_2$ and $t_{m+2}$ on $P_m$ with $\hat{f} = 4/(2 + \varepsilon')$ . One optimal schedule is shown in figure 3.3 . $f^* = 1.5$ . Hence, $\hat{f}/f^* = \frac{8}{6+3\varepsilon'} \to 4/3$ as $\varepsilon' \to 0$ . ■

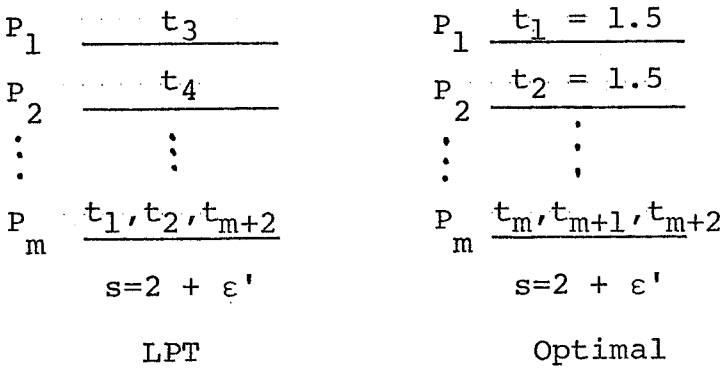



Figure 4.3.3 LPT and Optimal Schedules for Lemma 4.3.7

Lemma 4.3.8 For $m = 4$ and $5$ , $\hat{f}/f^* \leq 4/3$

Proof We prove only the case $m = 4$ . The proof for $m = 5$ is very similar and does not use any new techniques. The proof for $m = 4$ is by cases on the possible values of $n$ and $s$ . In what follows, we assume that the smallest set of jobs for which $\hat{f}/f^* \geq 4/3$ is of size $n$ and then arrive at a contradiction for all values of $n$ .

case a $s \geq 3$ . Substituting in Lemma 4.3.1 we obtain, for $s \geq 3$ and $m = 4$ , $\hat{f}/f^* \leq \frac{3+s}{1.5+s} \leq \frac{6}{4.5} = 4/3$ .

case b $n \leq 4$ there is either only one job on each of the four processors are idle in the optimal schedule . In the first case $\hat{f}/f^* = 1$ , in the second $\hat{f}_4/f_4^* \leq \hat{f}_3/f_3^* \leq 4/3$ .

case c $n = 5$ In both the LPT and optimal schedule there is job $t_i$ schedule alone on one of $P_1$-$P_3$ . Lemma 4.3.5 applies and $\hat{f}_4/f_4^* \leq \hat{f}_3/f_3^* \leq 4/3$ .

case d $n = 6$ $1.5 \leq s < 3$ Lemma 4.2.2 yields $\hat{f}/f^*$ $\leq 1 + \frac{3t_n}{(3+s)f^*}$ . By the assumption on the set of jobs $\hat{f}/f^* > 4/3$ . So, $\frac{3t_n}{(3+s)f^*} > 1/3$ or $f^* < \frac{9}{3+s} t_n$ $\leq 2t_n$ . The number of jobs on $P_1$ -$P_3$ is thus restricted to 1 and the number on $P_4$ is restricted to $\leq 4$ . The total number of jobs, $n$ , must be $\leq 7$ . When $n = 2m - 2 = 6$ , the proof of Lemma 4.3.6 applies as the optimal has at most one job on each of $P_1$-$P_3$ and $s \leq m-1$ . Hence, $\hat{f}_4/f_4^* \leq \hat{f}_3/f_3^* \leq 4/3$ . $s < 1.5$: (i) $t_1 \notin P_4$ in optimal. There must be at least 2 jobs on $P_4$ as otherwise we may interchange the job on $P_4$ with $t_1$ without increasing the finish time. So, at least two processors in the optimal schedule have only one job each. We may assume these jobs to be $t_1$ and $t_2$ . Since $s < 2$ , $t_2$ is in $P_1$ and alone in the LPT

schedule. Lemma 4.3.5 now applies and $\hat{f}/f^* \leq 4/3$ .

(ii) $t_1 \in P_4$ and $t_2 \notin P_4$ in optimal. Lemma 4.3.5 again applies.

(iii) $t_1 \in P_4$ and $t_2 \in P_4$ in optimal. Now. either Lemma 4.3.5 applies or $\hat{f} = (t_1+t_6)/s \leq f^*$ .

<u>case e</u> $n = 7$ : $1.5 \leq s < 3$ From case d we know that in the optimal each of $P_1$-$P_3$ has exactly one job scheduled on it while there are 4 jobs scheduled on $P_4$ . We examine all the possibilities.

(i) If $t_1 \notin P_4$ in the optimal then the optimal may be assumed to be:

$P_1$ $t_1$

$P_2$ $t_2$

$P_3$ $t_3$

$P_4$ $t_4, t_5, t_6, t_7$

In the LPT schedule jobs $t_2$ and $t_3$ cannot be alone on $P_1$ , $P_2$ or $P_3$ as, then Lemma 4.3.5 would apply and $\hat{f}_4/f_4^* \leq \hat{f}_3/f_3^*$ . Also, if $t_1$ is the only job on $P_4$ in the LPT schedule, then $\hat{f} \leq (t_1 + t_7)/s$ while $f^* \geq t_1$ . So, $\hat{f}/f^* \leq (t_1 + t_7)/(st_1) \leq 2/s \leq 4/3$ . This takes care of all possible LPT schedules with 7 jobs.

(ii) $t_1 \in P_4$ in the optimal and $t_2 \notin P_4$. This is very similar to (i). Unless in the LPT schedule $t_1$ is the only job scheduled on $P_4$, Lemma 4.3.5 applies and $\hat{f}/f^* \leq 4/3$. If $t_1$ is the only job on $P_4$ then $\hat{f} \leq (t_1 + t_7)/s$ while $f^* \geq (t_1 + t_5 + t_6 + t_7)/s \geq \hat{f}$.

The only remaining possibility is :

(iii) $t_1 \in P_4$ and $t_2 \in P_4$ in optimal. Now, $\hat{f} \leq (t_1 + t_2 + t_6 + t_7)/s$ for all possible LPT schedules, while $f^* \geq (t_1 + t_2 + t_6 + t_7)/s$.

$\underline{s < 1.5}$: (i) if $t_1 \notin P_4$ and $t_2 \notin P_4$ in optimal then $f^* \geq t_2 + t_7$ case d $f^* < \frac{9}{3+s} t_n$ => no more than two jobs on each of $P_1 - P_3$. But, $\hat{f} \leq t_2 + t_7$ as there are only 7 jobs.

(ii) $t_1 \notin P_4$ and $t_2 \in P_4$ in optimal => $f^* \geq \max\{t_1, \frac{t_2 + t_7}{s}\}$ since, $\hat{f} \leq t_2 + t_7$ $\hat{f}/f^* \leq s$ and so $s$ must be $> 4/3$ if $\hat{f}/f^*$ is to be $> 4/3$.

$\underline{4/3 < s < 1.5}$:

If $t_1$ is alone or with $t_7$ only on $P_4$ in the LPT schedule then $\hat{f} \leq (t_1 + t_7)/s \leq 3f^*/(2s) \leq 9f^*/8$

So, $t_1$ must be paired with a job other than $t_7$. Hence, $\hat{f} \leq t_3 + t_7$

if $t_3 \notin P_4$ in optimal then $f^* \geq t_3 + t_7$

if $t_3 \in P_4$ in optimal then $f^* \geq (t_2 + t_3)/s$

$\Rightarrow t_3 \leq s/2f^* \Rightarrow \hat{f} \leq (\frac{s}{2} + \frac{1}{2}) f^* \leq 1.25f^*$ ($t_7 \leq f^*/2$ as with $s \leq 1.5$ the number of jobs on $P_4$ must be less than 3 )

(iii) $t_1 \in P_4$ in the optimal

If $t_1$ is alone in the optimal then $\hat{f}/f^*$ is no worse than for identical processors. So, $\hat{f}/f^* \leq 4/3$ (see [13]) .

If $t_1$ is alone in the LPT schedule or coupled only with $t_7$ then $\hat{f} \leq (t_1 + t_7)/s \leq f^*$ . So, $\hat{f}$ must be $\leq t_3 + t_7$ .

If $t_2 \in P_4$ in optimal then $f^* \geq (t_1 + t_2)/s \geq 2t_2/s$ but $\hat{f} \leq t_3 + t_7 \leq (\frac{s}{2} + \frac{1}{2}) f^* \leq 1.25f^*$ .

If $t_2 \notin P_4$ in optimal then $t_2$ is alone on $P_1$ . In the LPT, since , $t_1$ is not alone on $P_4$ , $t_2$ must be alone on $P_1$ . So, Lemma 4.3.5 applies and $\hat{f}/f^* \leq 4/3$ .

case f $n = 8$

If $s \geq 1.5$ then case d requires at most 1 job on each of $P_1$-$P_3$ in optimal and at most 4 jobs on $P_4$. So, $n \leq 7$.

$s \leq 1.5$

(i) If $t_1 \notin P_4$ in optimal then since there can be at most 2 jobs on each processor, $f^* \geq t_1 + t_8$. Using the technique of case d we get for $\hat{f}/f^* > 4/3$, $f^* < \frac{9}{3+s} t_8 \leq (9/4)t_8$ or $t_8 > (4/9)f^*$. Hence, $t_1 < (5/9)f^*$. If $t_1$ is the only job on $P_4$ in the LPT schedule or $t_1$ and $t_8$ are the only jobs on $P_4$ then $\hat{f} \leq (t_1 + t_8)/s \leq f^*$. Hence, $\hat{f} \leq t_2 + t_8 \leq 2t_1 \leq (10/9)f^*$.

(ii) $t_1 \in P_4$ in optimal.

$f^* \geq (t_1 + t_8)/s$ and so for $\hat{f}/f^* > 4/3$ there must be a job other than $t_1$ and $t_8$ on $P_4$ in the LPT schedule. This implies $\hat{f} \leq t_2 + t_8 \leq f^*$ if $t_2 \notin P_4$ in optimal. Assume now that both $t_1$ and $t_2$ are on $P_4$ in the optimal. Then $f^* \geq (t_1 + t_2)s \geq 2t_2/s \Rightarrow t_2 \leq s/2f^*$. This, together with the knowledge that $t_8 \leq f^*/2$ and $\hat{f} \leq t_2 + t_8$ results in $\hat{f} \leq (\frac{s}{2} + \frac{1}{2})f^* \leq 1.25f^*$.

<u>case g</u> $n > 8$ Substittuting this into Lemma 4.2.2 yields

$$\hat{f}/f^* \leq 1 + \frac{m-1}{n} \leq 1 + (3/9) = 4/3 .$$

This takes care of all the possibilities and so for $\hat{f}_4/f_4^* \leq 4/3$ . ■

<u>Conjecture</u> $\hat{f}/f^* \leq 4/3$ for $m \geq 3$ and $s_i = 1$ , $1 \leq i < m$ and $s_m \geq 1$ .

# CHAPTER V

# OPEN SHOP

## 5.1 Introduction

A shop consists of $m \geq 1$ processors ( or machines). Each of these processors performs a different task. There are $n \geq 1$ jobs. Each job $i$ has $m$ tasks. The processing time for task $j$ of job $i$ is $t_{j,i}$. Task $j$ of job $i$ is to be processed on processor $j$, $1 \leq j \leq m$. A schedule for a processor $j$ is a sequence of tuples $(\ell_i, s_{\ell_i}, f_{\ell_i})$, $1 \leq i \leq r$. The $\ell_i$ are job indexes, $s_{\ell_i}$ is the start time of job $\ell_i$ and $f_{\ell_i}$ is the finish time. Job $\ell_i$ is processed continuously on processor $j$ from $s_{\ell_i}$ to $f_{\ell_i}$. The tuples in the schedule are ordered such that $s_{\ell_i} < f_{\ell_i} \leq s_{\ell_{i+1}}$, $1 \leq i < r$. There may be more than one tuple per job and it is assumed that $\ell_i \neq \ell_{i+1}$, $1 \leq i < r$. It is also required that each job $i$ spends exactly $t_{j,i}$ total time on processor $j$. A schedule for a m-shop is a set of $m$ processor schedules. One for each processor in the shop. In addition, these $m$ processor schedules must be such

that no job is to be processed simultaneously on two or more processors. A shop schedule will be abbreviated to schedule in future references. The finish time of a schedule is the latest completion time of the individual processor schedules and represents the time at which all tasks have been completed. An optimal finish time (OFT) schedule is one which has the least finish time amongst all schedules. A non-preemptive schedule is one in which the individual processor schedules has at most one tuple $(i, s_i, f_i)$ for each job $i$ to be scheduled. For any processor, $j$, this allows for $t_{j,i} = 0$ and also requires that $f_i - s_i = t_{j,i}$. A schedule in which no restriction is placed on the number of tuples per job per processor is preemptive. Note that all non-preemptive schedules are also preemptive while the reverse is not true.

Open shop schedules differ from flow shop and job shop [5,7] schedules in that in an open shop, no restrictions are placed on the order in which the tasks for any job are to be processed. In this Chapter we shall investigate OFT schedules for the open shop. It is clear that when $m = 1$, OFT schedules can be trivially obtained. We shall therefore restrict ourselves to the case $m > 1$. First, in section 5.2 we show that preemptive and nonpreemptive OFT schedules can be obtained in linear time when $m = 2$. This

contrasts with Johnson's O(n log n) algorithm [7, p89] for the 2 processor flow shop. When $m > 2$ OFT preemptive schedules can still be obtained in polyno - mial time (section 5.3).

For nonpreemptive scheduling, however, finding OFT schedules when $m > 2$ is NP-Complete. These results may be compared to similar results obtained for flow shop and job shop OFT scheduling. In [11] and in Chap ter VI it is shown that finding nonpreemptive OFT schedules for the flow shop when $m > 2$ and the job shop when $m > 1$ are NP-Complete. In Chapter VI it is also shown that finding preemptive OFT schedules for the 3 processor flow shop and 2 processor job shop are NP-Complete. Thus, as far as the complexity of finish time scheduling is concerned, open shops are easier to schedule when a preemptive schedule is desired.

## 5.2 OFT Scheduling for m = 2

In this section, a linear time algorithm to obtain a nonpreemptive and preemptive OFT schedule for the case of two processors is presented. For notational simplicity, we denote $t_{1,i}$, the task time on proces- sor 1, by $a_i$ and $t_{2,i}$ by $b_i$, $1 \leq i \leq n$. Infor - mally, the algorithm proceeds by dividing the jobs into two groups A and B. The jobs in A have $a_i \geq b_i$ while those in B have $a_i < b_i$. The schedule is

build from the "middle" with jobs from A being added on at the right while those from B are added on at the left. The schedule from the jobs in A is such that there is no idle time on processor 1 (except at the end) and for each job in A, it is possible to start its execution on processor 2 immediately following its completion on processor 1. The part of the schedule made up with jobs in B is such that the only idle time on processor 2 is at the beginning. In addition, the processing of a job on processor 1 can be started such that its processing on processor 2 can be carried out immediately after completion on processor 1. Finally, some finishing touches involving only the first and last jobs in the schedule are made. This guarantees an optimal schedule.

line no.

```
1   Algorithm OPENSHOP
    // This algorithm finds a minimum finish time non-
       preemptive schedule for the open shop problem
       with task times (a_i ; b_i), 1 ≤ i ≤ n
       Initialize variables: a_0;b_0 represent a dummy
                                      job
                             T_i = sum of task times
                                   assigned to processor
                                   i,  1 ≤ i ≤ 2.
```

```
                                  ℓ = index of leftmost job in
                                      the schedule
                                  r = index of rightmost job in
                                      the schedule.
                                  S_i = sequence for processor i
                                       1 ≤ i ≤ 2 //
2   T_1 ← T_2 ← a_0 ← b_0 ← ℓ ← r ← 0 ; S ← null
    // schedule the n jobs //
3   for i ← 1 to n do
4      T_1 ← T_1 + a_i ; T_2 ← T_2 + b_i
5      if a_i ≥ b_i then [ if a_i ≥ b_r then
                                  [ // put r on right, || means
                                      string concatenation //
6                                 S ← S || r ; r ← i ]
                                      else  [// put i on right//
7                                 S ← S || i ]]
8                  else [ if b_i ≥ a_ℓ then
                                  [ // put ℓ on left //
9                                 S ← ℓ || S ; ℓ ← i ]
                                      else
                                  [ // put i on left //
10                                S ← i || S ]]
11  end   //now start finishing touch  //
12  delete all occurrances of job 0 from S
13  if T_1 - a_ℓ < T_2 - b_r then [ S_1 ← S || r || ℓ;
                                       S_2 ← ℓ || S || r ]
```

14 else [ $S_1 \leftarrow \ell \,||\, S \,||\, r$ ;
$S_2 \leftarrow r \,||\, \ell \,||\, S$ ]

// an optimal schedule is obtained by processing jobs on processor i in the order specified by $S_i$, $1 \leq i \leq 2$. The exact schedule may be determined using Theorem 5.2.1 //

15 return

16 end of OPENSHOP ■

Example 5.2.1 Consider the open shop problem with 6 jobs having task times as below:

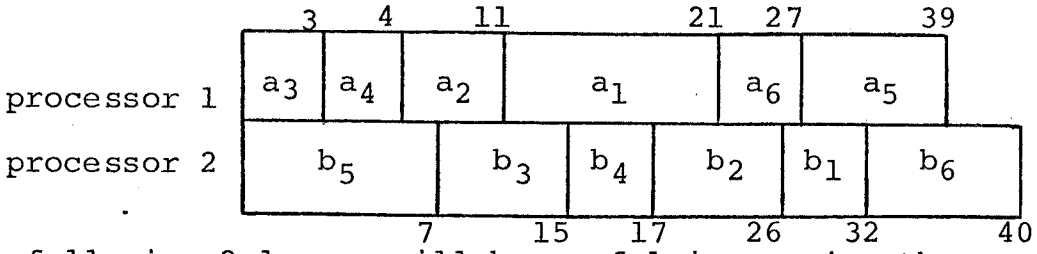
| Processor \ Job | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 10 | 7 | 3 | 1 | 12 | 6 |
| 2 | 6 | 9 | 8 | 2 | 7 | 8 |

Initially, $\ell = r = 0$ and $S = \emptyset$. The following table gives the values of S, r, $\ell$ at the end of each iteration of the for loop 3-12.

| End of iteration | S | r | $\ell$ |
|---|---|---|---|
| 1 | 0 | 1 | 0 |
| 2 | 00 | 1 | 2 |
| 3 | 200 | 1 | 3 |
| 4 | 4200 | 1 | 3 |
| 5 | 42001 | 5 | 3 |
| 6 | 420016 | 5 | 3 |

After deleting the 0's from S we have S = 4216, r = 5,

$\ell = 3$, $T_1 = 39$ and $T_2 = 40$. Since $T_1 - a_3 > T_2 - b_5$ we get $S_1 = 342165$ and $S_2 = 534216$. Processing by these permutations gives the Gantt chart:




The following 2 lemmas will be useful in proving the correctness of algorithm OPENSHOP.

Lemma 5.2.1 Let the set of jobs being scheduled be such that $a_i \geq b_i$, $1 \leq i \leq n$ and let $D$ be the permutation obtained after deleting the 0's from $S$ in line 12 of algorithm OPENSHOP and concatenating $r$ to the right. The jobs 1-n may be scheduled in the order $D$ such that:

(i) there is no idle time on processor 1 except following the completion of the last task on this processor

(ii) For every job $i$, its processor 1 task is completed before the start of its processor 2 task

(iii) for last job $r$, the difference, $\Delta$, between the completion time of task 1 and the start time of task 2 is zero.

Proof The proof is by induction on $n$. The lemma is clearly true for $n = 1$. Assume that the lemma is true

for $1 \leq n < k$. We shall show that it is also true for $n = k$. Let the k jobs be $J_1, J_2, \ldots, J_k$ and let $r'$ be the value of $r$ at the beginning of the iteration of the "for" loop of lines 3-11 when $i = n$. From the algorithm it is clear that the per - mutation, $D'$, obtained at line 12 when the $k-1$ jobs $J_1, J_2, \ldots, J_{k-1}$ are to be scheduled is of the form $D"r'$. Moreover, $D = D"r'k$ or $D = D"kr'$. From the induction hypothesis, it follows that tha jobs $J_1, J_2, \ldots, J_{k-1}$ can be scheduled according to the permutation $D"r'$ so as to satisfy (i) - (iii) of the lemma. I.e., these k-1 jobs may be scheduled as in figure 5.2.1 . Let $i$ be the job immediately preceding $r'$ in $D'$. In case $k = 2$, let $i = 0$ with $a_0 = b_0 = 0$.

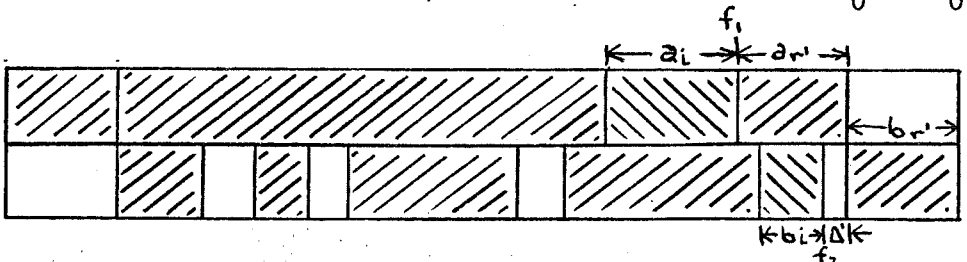



Figure 5.2.1 Scheduling by $D' = D"r'$ indicate task processing. Last job is $r'$. $\Delta' \geq 0$ .

If $A_k \geq b_{r'}$ then $D = D'k$ and it is clear the job $k$ can be added on to the schedule of figure 5.2.1 at the right end, so that (i) - (iii) of the lemma hold.

If $A_k < b_{r'}$ then $D = D"kr'$. Now, job $r'$ is

moved $a_k$ units to the right so that $a_k$ can be accomodated between $i$ and $r'$ satisfying (i). Let $f_1$ be the finish time of $a_i$ and $f_2 \geq f_1$ be the finish time of $b_i$. The finish time of $a_k$ is then $f_1 + a_k < f_1 + a_{r'}$ as $a_{r'} \geq b_{r'}$. By (iii) the start time of $b_{r'}$ has to be $f_1 + a_k + a_{r'}$. Also, we know, from the induction hypothesis, that $f_1 + a_{r'} - f_2 = \Delta' \geq 0$. I.e. $f_1 + a_{r'} \geq f_2$. The earliest that $b_k$ may be scheduled is $\max\{f_1 + a_k , f_2\} < f_1 + a_{r'}$. This implies that there is enough time between the start time of $b_{r'}$ and the earliest start time of $b_k$ to complete the processing of $b_k$. ■

Lemma 5.2.2 Let the set of jobs being scheduled be such that $a_i < b_i$, $1 \leq i \leq n$ and let $C$ be the permutation obtained after deleting the 0's from $S$ in line 12 of algorithm OPENSHOP and concatenating $\ell$ to the left. The jobs may be scheduled in the order $C$ such that:

(i) there is no idle time on processor 2 except at the beginning

(ii) for every task $i$, its processor 1 task is completed before the start of its processor 2 task.

(iii) for the first job, $\ell$, the difference, $\Delta$ between the completion time of task 1 and

the start time of task 2 is zero.

Proof The proof is similar to that of Lemma 5.2.1. ■

Lemma 5.2.3 Let $(a_i, b_i)$ be the processing times for job $i$ on processors 1 and 2 respectively, $1 \leq i \leq n$. Let $f^*$ be the finish time of an optimal finish time preemptive schedule. Then, $f^* \geq \max\{\max_i\{a_i + b_i\}, T_1, T_2\}$ where $T_1 = \sum_1^n a_i$ and $T_2 = \sum_1^n b_i$.

Proof Obvious. ■

We are now ready to prove the correctness of algorithm OPENSHOP.

Theorem 5.2.1 Algorithm OPENSHOP generates optimal finish time schedules.

Proof Let $J_1, J_2, \ldots, J_n$ be the set of jobs being scheduled. Let $A$ be the subset with $a_i \geq b_i$ and $B$ be the remaining jobs. It is easy to verify that the theorem is true when either $A$ or $B$ is empty. So, assume $A$ and $B$ to be nonempty sets. Let $E$ be the permutation obtained after deleting the 0's from $\ell||S||r$ in line 12. Then $E = CD$ where $C$ consists solely of jobs in $B$ and $D$ consists solely of jobs in $A$. From Lemmas 5.1.1 and 5.1.2, it follows that

the jobs in A and B may be scheduled in the orders D and C to obtain schedules as in figure 5.2.2. In the schedules of figure 5.2.2, the processor 1 tasks for C and the processor 2 tasks for D have to be scheduled such that all the idle time appears either at the end or at the beginning.

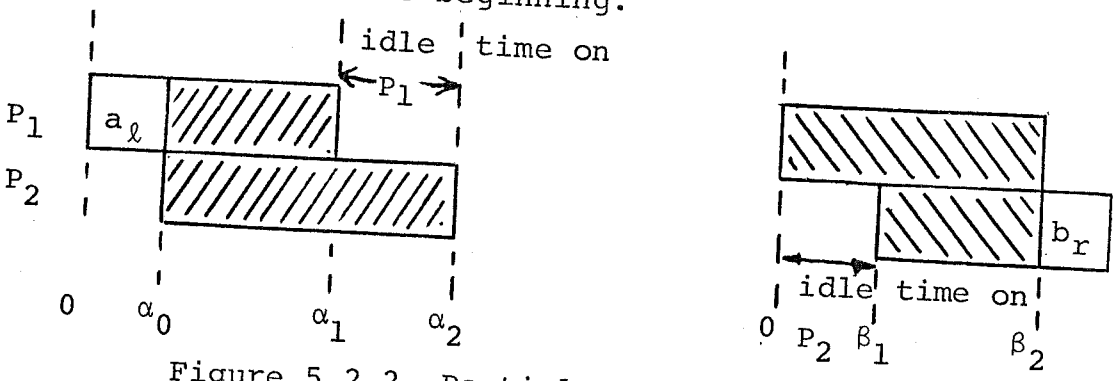



Figure 5.2.2 Partial schedules obtained for sets B and A respectively.

Let $T_1 = \sum_1^n a_i$ and $T_2 = \sum_1^n b_i$ . The schedule for the entire set of jobs is obtained by merging the two schedules of figure 5.2.2 together so that either

(a) The blocks on P1 meet first. This happens when $(\alpha_2 - \alpha_1) \leq \beta_1$ .

or (b) The blocks on P2 meet first. This happens when $(\alpha_2 - \alpha_1) > \beta_1$ .

Let us consider these two cases separately.

<u>case a</u> $\alpha_2 - \alpha_1 \leq \beta_1$

This happens when $T_1 - a_\ell \geq T_2 - b_r$ . In this case, line 14 of the algorithm results in the tasks on P1 being processed in the order CD while those on

P2 are processed in the order rCD' where D' is D with r deleted. The section $\alpha_0 - \alpha_2$ of figure 5.2.2 (a) is now shifted right until it meets with $\beta_1 - \beta_2$ of figure 5.2.2 (b). Task $b_r$ is moved to the leftmost point. The finish time of the schedule obtained becomes $\max\{a_r + b_r, T_1, T_2\}$ which by Lemma 5.2.3 is optimal.

case b $\alpha_2 - \alpha_1 > \beta_1$

This happens when $T_1 - a_\ell < T_2 - b_r$. In this case, line 13 of the algorithm results in the tasks on P1 being processed in the order C'D$\ell$ where C' is C with $\ell$ deleted. Tasks on P2 are processed in the order CD. The schedule is obtained by processing tasks on P2 with no idle time starting at time 0. Tasks on P1 are processed with no idle time (except at the end) in the order C'D. Task $a_\ell$ is started as early as possible following C'D. The finish time is seen to be $\max\{a_\ell + b_\ell, T_1, T_2\}$ which by Lemma 5.2.3 is optimal.

This completes the proof. ■

Corollary 5.2.1 Algorithm OPENSHOP generates optimal preemptive schedules for $m = 2$.

Proof Follows immediately from the proof of Theorem 5.2.1 . ■