CHAPTER FIVE

BOUNDS ON THE PERFORMANCE OF SCHEDULING ALGORITHMS

R. L. GRAHAM
BELL LABORATORIES, MURRAY HILL, NEW JERSEY

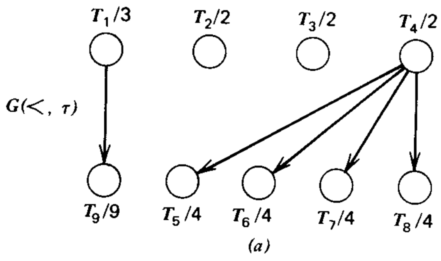
In this chapter we investigate the worst-case behavior of a number of scheduling algorithms for the general multiprocessor-with-resources model, as well as numerous important special cases. The model and notation are defined in Chapter 1. We begin in Section 5.1 by studying, under the framework of list scheduling, the rather unpredictable dependence of the schedule length ω on the various parameters of the problem, even for the case of no additional resource constraints. In Sections 5.2 and 5.3 the performance of critical path scheduling is examined, and in Section 5.4 bounds for the extended model including additional resource constraints are derived. In Section 5.5 heuristics are covered for the (bin-packing) problem of minimizing the number of processors to meet a given deadline. Finally, in Section 5.6, bounds for a number of related problems are presented. The complexity of the various problems studied in this chapter has been analyzed in the preceding chapter.

5.1 MULTIPROCESSOR SCHEDULING ANOMALIES

We begin with an example that illustrates, using a single-task system, the anomalies that can arise in varying any one of the parameters, including the priority list.

Example 1 The graph $G(<, \tau)$ of a task system appears in Fig. 5.1*a*. Holding the parameters fixed, we see in Figs. 5.2 through 5.5, the effects of changing the priority list, increasing the number of processors, reducing execution times, and weakening the precedence constraints, respectively. The basis for comparison is the optimal schedule in Fig. 5.1*b*.

Note that in Figs. 5.3, 5.4, and 5.5, although we would intuitively expect that the changes made would cause ω to decrease, in fact, an *increase* in ω occurred.



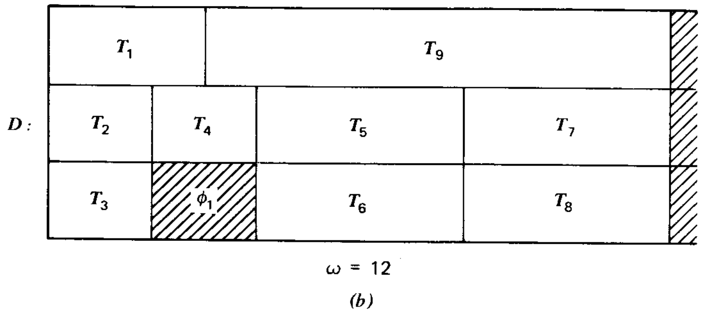


Figure 5.1 A task system and optimal schedule. (*a*) $m = 3$, $L = (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9)$. Figures 5.2 through 5.5 indicate effects of changes in L, m, τ, and $<$.

The next example shows that such an increase in ω is not necessarily caused by a poor choice of the list L but in fact is inherent in the model itself.

Example 2 Figure 5.6*a* shows a task system for which we assume $m = 2$ and the list as shown. An optimal schedule appears in Fig. 5.6*b*. Now consider the same task system except for execution times given by the new function $\tau' = \tau - 1$. We find that *no matter what list is assumed* for

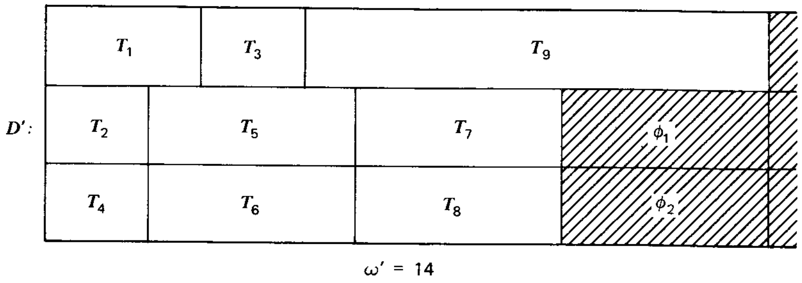


Figure 5.2 Priority list changed: L becomes $L' = (T_1, T_2, T_4, T_5, T_6, T_3, T_9, T_7, T_8)$.

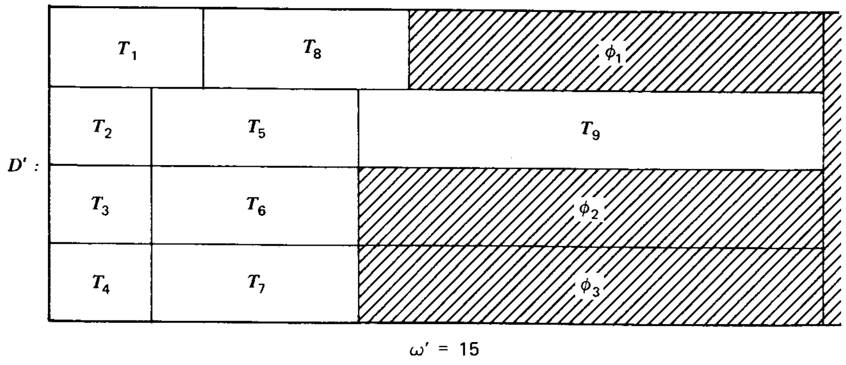


Figure 5.3 Number of processors increased: m is changed to $m' = 4$.

the new system, we cannot obtain a schedule whose length is less than that in Fig. 5.7.

We now derive a general bound on the relative effects on schedule length of changes in one or more problem parameters. Suppose we are given a set $\mathcal{T}$ of tasks, which we execute twice. The first time we use an execution time function τ, a partial order $<$, a priority list L, and a system composed of m identical processors. The second time we use a time function $\tau' \leq \tau$, a partial order $<' \subseteq <$, a priority list L' and a system composed of m' identical processors. As usual, ω and ω' denote the corresponding finishing times.

Theorem 5.1 [G1] Under the assumptions already stated, we have

$$\frac{\omega'}{\omega} \leq 1 + \frac{m-1}{m'}$$

Proof Consider the timing diagram D' obtained by executing the tasks T_i of $\mathcal{T}$ using the primed parameters. Define a partition of $[0, \omega')$ into two

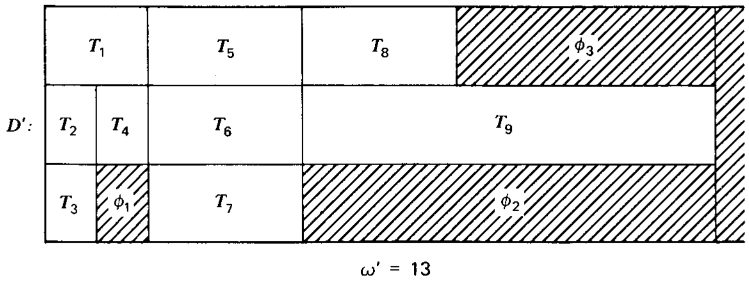
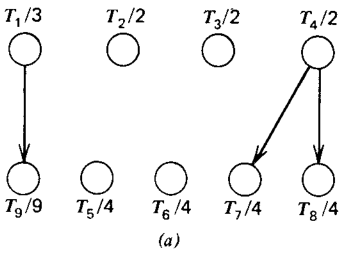


Figure 5.4 Execution times reduced: τ is changed to $\tau' = \tau - 1$.



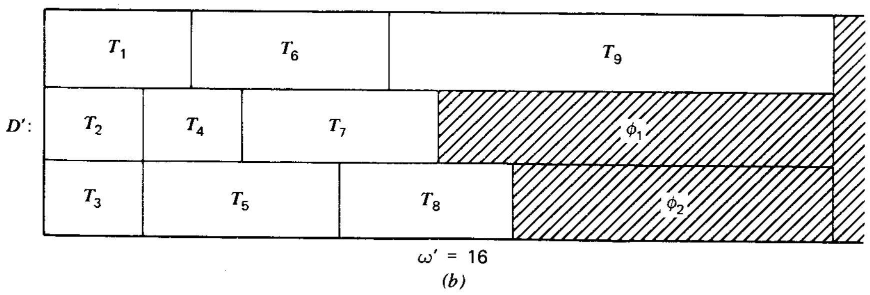
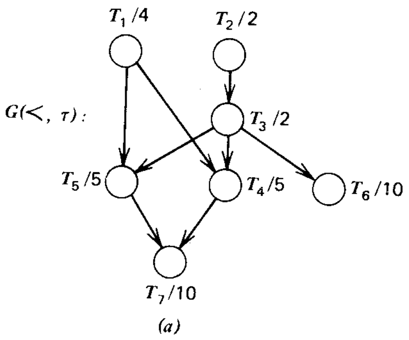


Figure 5.5 Precedence constraints weakened: $<$ is changed to $<' = < - \{(T_4, T_5), (T_4, T_6)\}$. (a) $m = 3$, $L = (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9)$.



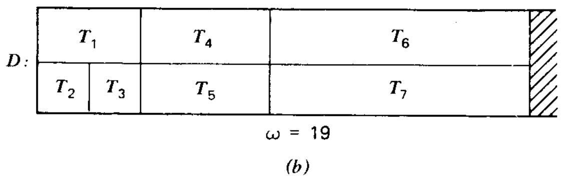


Figure 5.6 A task system and optimal schedule. (a) $m = 2$, $L = (T_1, T_2, T_3, T_4, T_5, T_6, T_7)$.

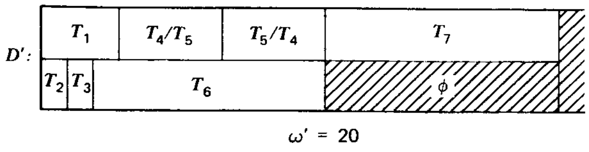


Figure 5.7 An optimal schedule assuming $\tau' = \tau + 1$.

subsets A and B as follows:

$$A = \{t \in [0, \omega') | \text{ all processors are busy at time } t\}$$
$$B = [0, \omega') - A$$

Note that A and B are both the unions of disjoint half-open intervals. Let T_{j_1} denote a task that finishes in D' at time ω' (i.e., such that $f_{j_1} = \omega'$). There are two possibilities.

1. If s_{j_1}, the starting time of T_{j_1}, is an *interior* point of B, then by the definition of B there is some processor P_i which for some $\varepsilon > 0$ is idle during the time interval $[s_{j_1} - \varepsilon, s_{j_1})$. The only way this can happen, however, is if for some task T_{j_2} we have $T_{j_2} <' T_{j_1}$ and $f_{j_2} = s_{j_1}$.

2. On the other hand, suppose s_{j_1} is not an interior point of B. Furthermore, suppose $s_{j_1} \neq 0$. Let $x_1 = \text{l.u.b.}\{x | x < s_{j_1} \text{ and } x \in B\}$, or 0 if the set is empty. By the construction of A and B, we see that $x_1 \in A$, and for some $\varepsilon > 0$, P_i is idle during the time interval $[x_1 - \varepsilon, x_1)$. But again, this can occur only because of some task $T_{j_2} <' T_{j_1}$ which is being executed during this time interval.

Thus we have seen that *either* there exists a task $T_{j_2} <' T_{j_1}$ so that $y \in [f_{j_2}, s_{j_1})$ implies $y \in A$ *or* we have $x < s_{j_1}$ implies either $x \in A$ or $x < 0$.

We can repeat this procedure inductively, forming $T_{j_3}, T_{j_4}, \ldots$, until we reach a task T_{j_r} for which $x < s_{j_r}$ implies either $x \in A$ or $x < 0$. Hence we have shown the existence of a chain of tasks

$$T_{j_r} <' T_{j_{r-1}} <' \cdots <' T_{j_2} <' T_{j_1} \tag{1}$$

such that in D' at every time $t \in B$, some T_{j_k} is being executed. This implies that

$$\sum_{\emptyset' \in D'} \tau'(\emptyset') \leq (m' - 1) \sum_{k=1}^{r} \tau'_{j_k} \tag{2}$$

where the sum of the left-hand side is taken over all empty tasks $\emptyset'$ in D'. But by (1) and the hypothesis $<' \subseteq <$ we have

$$T_{j_r} < T_{j_{r-1}} < \cdots < T_{j_2} < T_{j_1} \tag{3}$$

Therefore,

$$\omega \geq \sum_{k=1}^{r} \tau_{j_k} \geq \sum_{k=1}^{r} \tau'_{j_k} \tag{4}$$

Consequently, by (2) and (4), we have

$$\begin{aligned}\omega' &= \frac{1}{m'}\left\{\sum_{k=1}^{n} \tau'_k + \sum_{\emptyset' \in D'} \tau'(\emptyset')\right\} \\ &\leq \frac{1}{m'}(m\omega + (m'-1)\omega)\end{aligned} \tag{5}$$

From this we obtain

$$\frac{\omega'}{\omega} \leq 1 + \frac{m-1}{m'} \tag{6}$$

and the theorem is proved. ▣

The following examples show not only that the bound of Theorem 5.1 is best possible, but in fact it can be achieved (asymptotically) by varying any one *of the parameters.*

Example 3 In this example L varies, $<$ is empty, and m is arbitrary. The task/execution times are given by

$$T_1/1, T_2/1, \ldots, T_{m-1}/1, T_m/m-1, T_{m+1}/m-1, \ldots, T_{2m-2}/m-1, T_{2m-1}/m$$

Figure 5.8 presents the first list to be used and the resulting schedule. The second list and the resulting longer schedule are given in Fig. 5.9. As can be seen,

$$\frac{\omega'}{\omega} = 2 - \frac{1}{m}$$

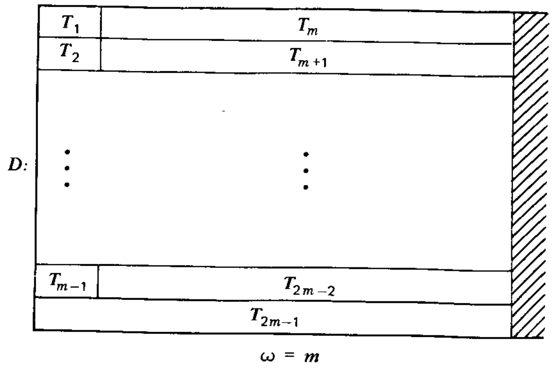


Figure 5.8 An optimal schedule: $L = (T_1, T_2, \ldots, T_{m-1}, T_m, T_{m+1}, \ldots, T_{2m-2})$.

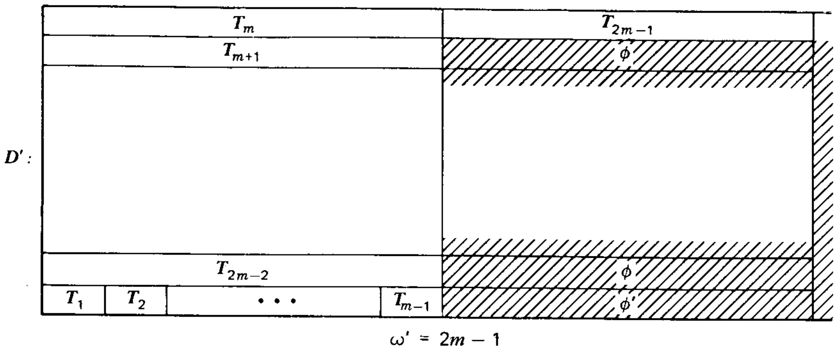


Figure 5.9 A bad schedule: $L' = (T_m, T_{m+1}, \ldots, T_{2m-2}, T_1, T_2, \ldots, T_{m-1}, T_{2m-1})$.

Example 4 In this example τ decreases. Here, as in the remainder of the chapter, ε denotes a suitably small positive number. A task system and corresponding optimal schedule are illustrated in Fig. 5.10. Figure 5.11 shows the effect of the following change in execution times

$$\tau' = \begin{cases} \tau_i - \varepsilon & \text{for } 1 \le i \le m-1 \\ \tau_i & \text{otherwise} \end{cases}$$

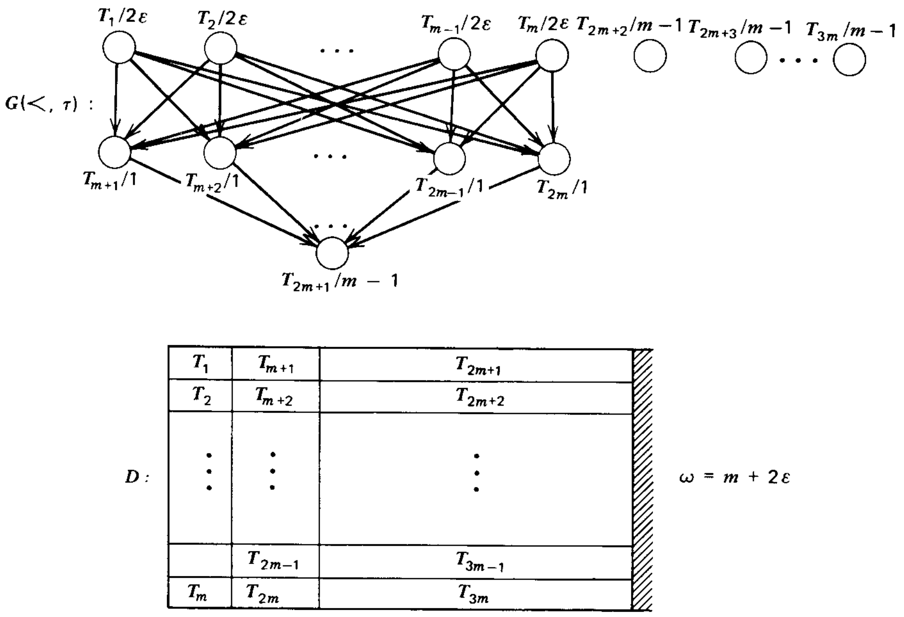


Figure 5.10 An optimal schedule.

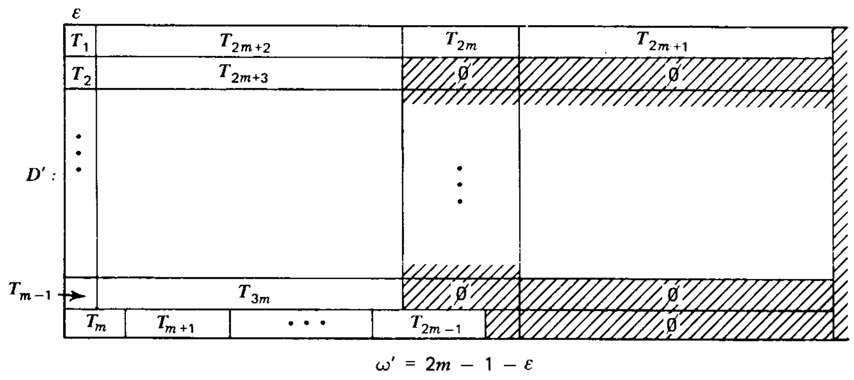


Figure 5.11 A bad schedule.

Inspecting the figures, we find

$$\frac{\omega'}{\omega} = \frac{2m-1+\varepsilon}{m+2\varepsilon} \to 2 - \frac{1}{m} \quad \text{as} \quad \varepsilon \to 0$$

Example 5 We weaken the precedence constraint $<$ in this example. Compare the task system and an optimal schedule (Fig. 5.12) with the results when all the precedence constraints are removed (Fig. 5.13). From the figures, we have

$$\frac{\omega'}{\omega} = \frac{2m-1}{m+\varepsilon} \to 2 - \frac{1}{m} \quad \text{as} \quad \varepsilon \to 0$$

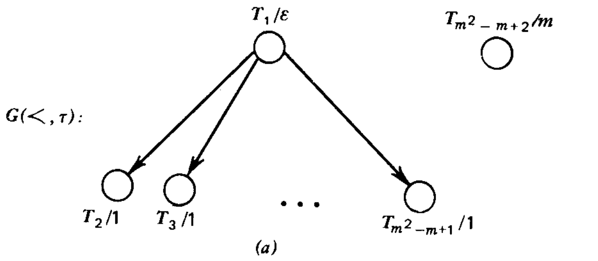
Example 6 Finally, we consider increases in the number of processors. Suppose we are given the task system represented in Fig. 5.14*a* whose optimal schedule on m processors appears in Fig. 5.14*b*. Now let $m' > m$. We obtain the longer schedule given in Fig. 5.14*c*. Forming the ratio of schedule lengths, we obtain

$$\frac{\omega'}{\omega} = \frac{m'+m-1+\varepsilon}{m'+2\varepsilon} \to 1 + \frac{m-1}{m'} \quad \text{as} \quad \varepsilon \to 0$$

For the case $m' < m$, a similar example exists, but we do not give it here.

The following example, due to M. Kaufman [K1], shows that (6) can be achieved by varying L, even if $<$ is a forest and all $\tau_i = 1$.

Example 7 Consider the graph shown in Fig. 5.15*a* and suppose we have unit execution times for all tasks. The optimal schedule and corresponding list is shown in Fig. 5.15*b*. If we change the list to that given in Fig.



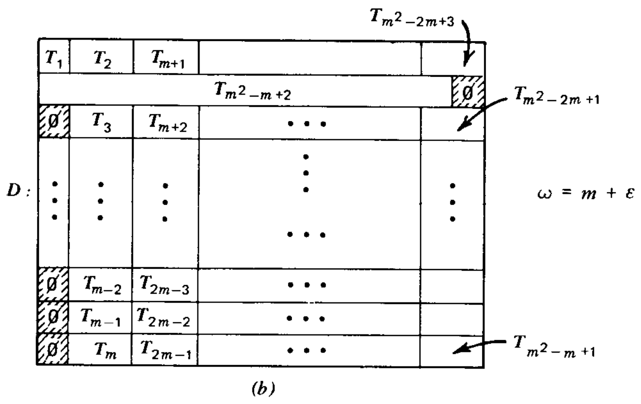


Figure 5.12 An optimal schedule. (*a*) $L = (T_1, T_2, \ldots, T_{m^2-m+2})$.

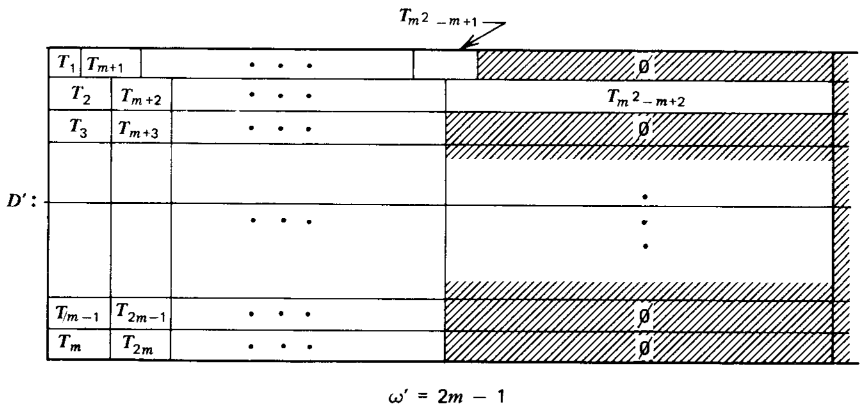


Figure 5.13 A bad schedule.

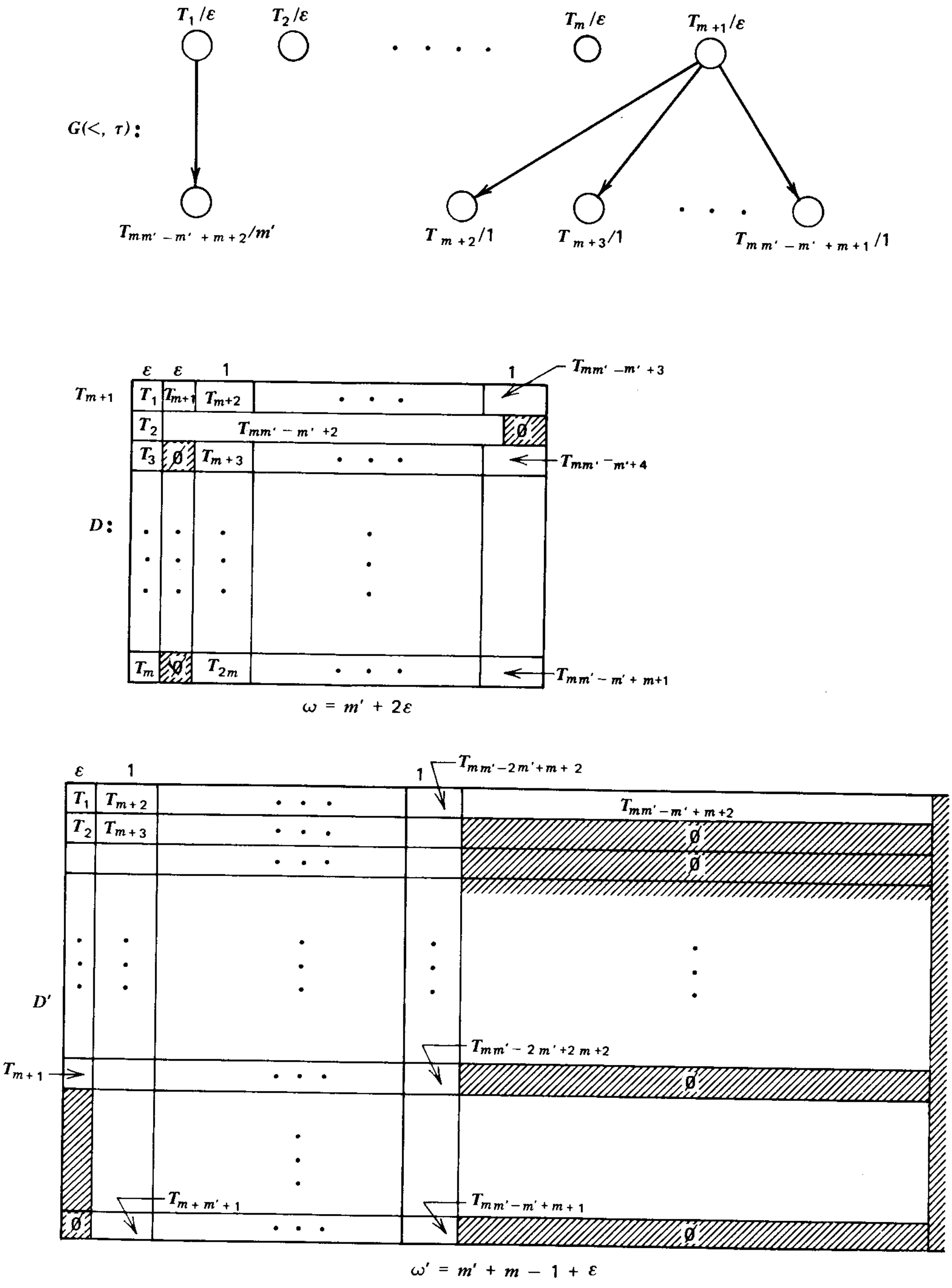
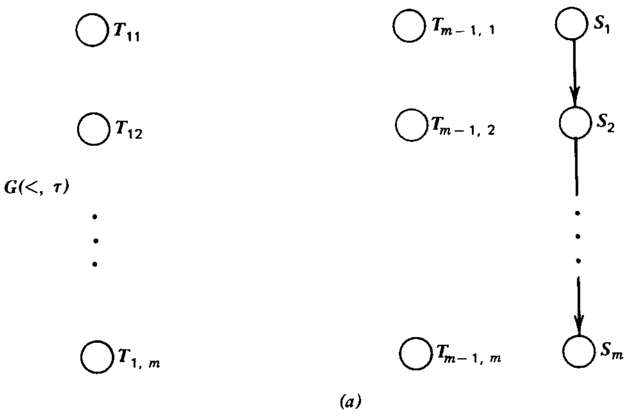


Figure 5.14 Task system (*a*) for whose optimal schedule *m* varies (*L*) and (*c*).



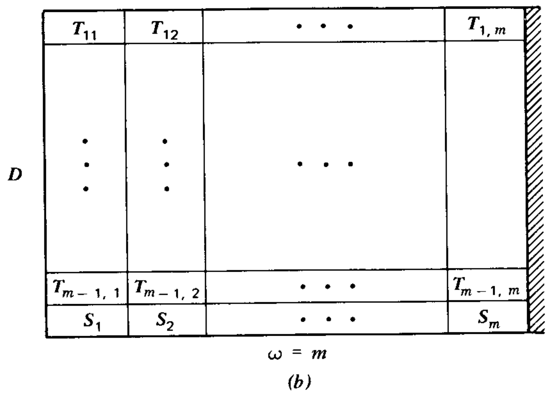


Figure 5.15 A task system and optimal schedule. (*a*) $L = (T_{11}, \ldots, T_{m-1,1}, S_1, T_{12}, \ldots, T_{m-1}, S_2, T_{21}, \ldots, S_m)$.

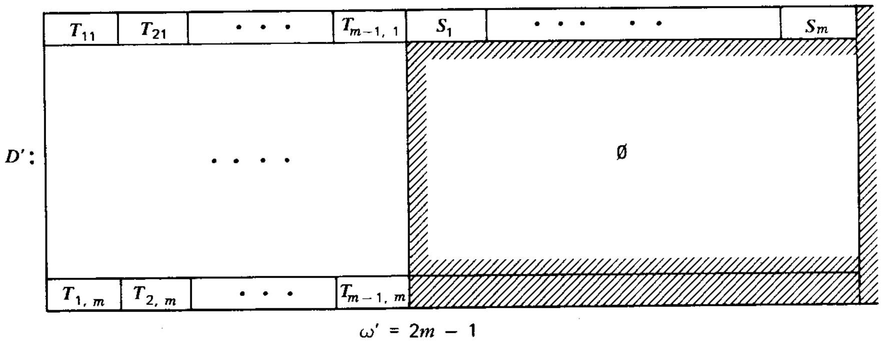


Figure 5.16 A very bad schedule. $L' = (T_{11}, T_{12}, \ldots, T_{1,m}, \ldots, T_{m-1,1}, \ldots, T_{m-1,m}, S_1, \ldots, S_m)$.

5.16 we obtain the longer schedule shown. Evidently,

$$\frac{\omega'}{\omega} = 2 - \frac{1}{m}$$

Finally, we give an example that again achieves the bound of $2 - 1/m$ by varying L, this time with $<$ empty and

$$\frac{\max\{\tau_i\}}{\min\{\tau_i\}} \leq 4$$

Whether the constant 4 is best possible here is not known. In any case, we see that the occurrence of the worst possible behavior does not depend on having tasks with widely disparate execution times.

Example 8 Let $<$ be empty and suppose the maximum ratio of execution times is no greater than 4. We consider three cases depending on $m \pmod 4$. For $m = 2r$, let $\tau = (\tau_1, \ldots, \tau_{2m+1})$ be given by

$$(r, r, r+1, r+1, \ldots, 2r-2, 2r-2, 2r-1, 2r-1,$$
$$3r-2, 3r-2, 3r-3, 3r-3, \ldots, 2r-1, 2r-1, 4r)$$

Then, by using the corresponding list $L = (T_1, \ldots, T_{2m+1})$, we obtain the schedule shown in Fig. 5.17*a*. Since the optimal schedule is as shown in Fig. 5.17*b*, when ω_0 is m, we have,

$$\frac{\omega_L}{\omega_0} = 2 - \frac{1}{2r} = 2 - \frac{1}{m}$$

For $m = 4r + 1$, *let* τ be given by

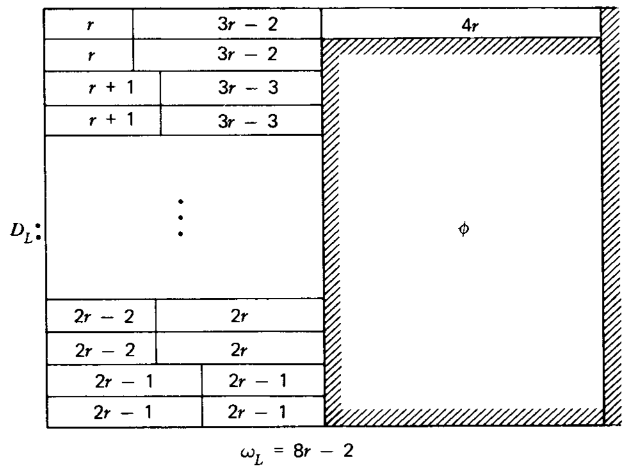
$$(2r+1,\ 2r+1,\ 2r+1,\ 2r+1,\ 2r+3,\ 2r+3,\ 2r+3,\ 2r+3, \ldots, 4r-1,$$
$$4r-1,\ 4r-1,\ 4r-1,\ 4r,\ 6r-1,\ 6r-1,\ 6r-1,\ 6r-1,\ 6r-3,\ 6r-3,$$
$$6r-3,\ 6r-3,\ \ldots,\ 4r+1,\ 4r+1,\ 4r+1,\ 4r,\ 8r+2)$$

In this case by using the list $L = (T_1, \ldots, T_{2m+1})$ we obtain the schedule of Fig. 5.18*a* whereas the optimal schedule is as shown in Fig. 5.18*b*. Again we obtain

$$\frac{\omega_L}{\omega_0} = 2 - \frac{1}{4r+1} = 2 - \frac{1}{m}$$

Finally, let $m = 4r + 3$ and let τ be given by

$$(r+1, r+1, r+1, r+1, r+2, r+2, r+2, r+2, \ldots, 2r+1, 2r+1, 2r+1,$$
$$3r+1, 3r+1, 3r+1, 3r+1, 3r, 3r, 3r, 3r, \ldots,$$
$$2r+2, 2r+2, 2r+2, 2r+2, 2r+1, 2r+1, 2r+1, 4r+3)$$



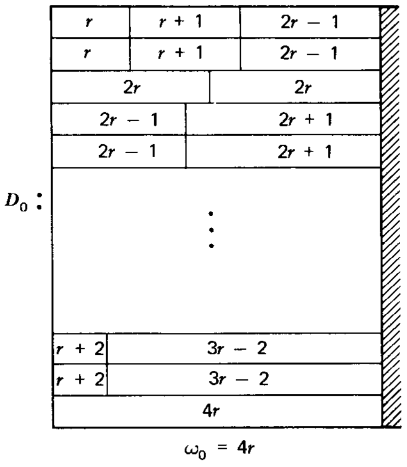
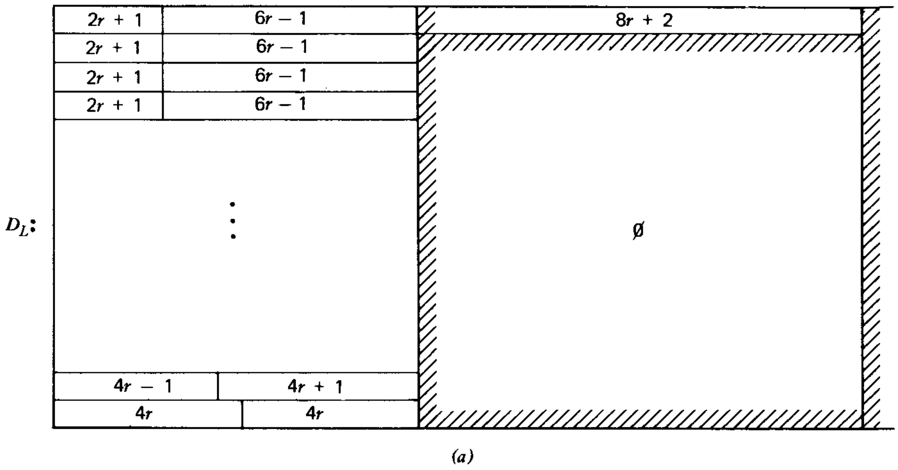


Figure 5.17 An extremal example for $m = 2r$.

The schedule in Fig. 5.19*a* is produced with these parameters using the list $L = (T_1, \ldots, T_{2m+1})$. The optimal schedule is given in Fig. 5.19*b* and shows that

$$\frac{\omega_L}{\omega_o} = 2 - \frac{1}{4r+3} = 2 - \frac{1}{m}$$



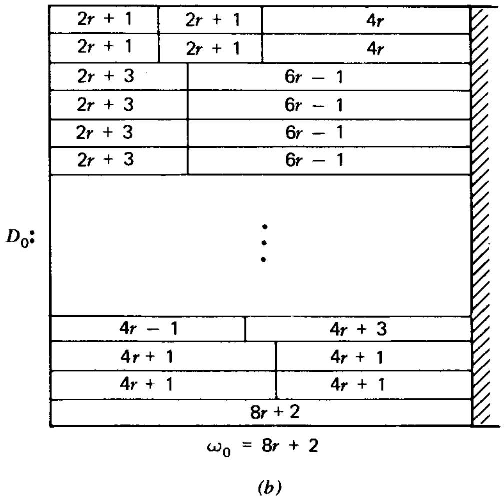
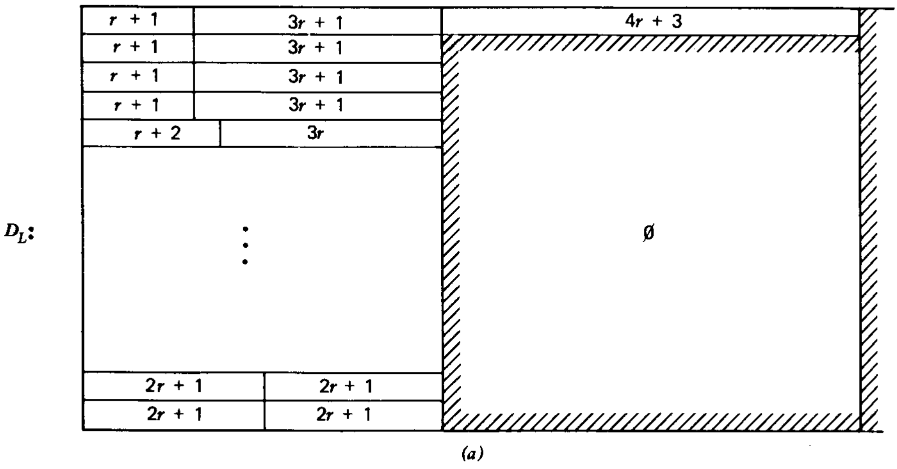


Figure 5.18 An extremal example for $m = 4r + 1$.

5.2 BOUNDS FOR INDEPENDENT TASKS AND NO ADDITIONAL RESOURCES

In this section we examine the special case in which the partial order $<$ is empty and, as before, $s = 0$ (i.e., there are no resource constraints). As shown in Example 3, a poor choice of the list L can still result in the worst



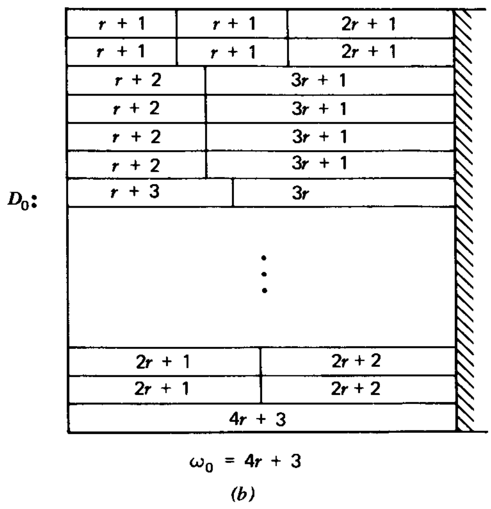


Figure 5.19 An extremal example for $m = 4r + 3$.

possible finishing time ω, that is, $\omega = (2 - 1/m)\omega_0$. However, if a little care is taken to prepare L, the bounds on ω can be improved considerably.

The first algorithm we consider for scheduling $\mathcal{T}$ is an example of a *critical path* algorithm (see Chapter 1). In this case, we simply form L by arranging the T_i in order of decreasing τ_i. Denote the corresponding finishing time by ω_{CP}.

Theorem 5.2 [G2] If $<$ is empty, then

$$\frac{\omega_{\mathrm{CP}}}{\omega_0} \leq \frac{4}{3} - \frac{1}{3m} \tag{7}$$

Furthermore, examples exist which achieve this bound.

Proof Assume $\mathcal{T} = \{T_1, \ldots, T_n\}$ is a set of tasks with times τ_i which contradicts (7). Furthermore, we can assume that the T_i have been labeled so that $\tau_1 \geq \tau_2 \geq \cdots \geq \tau_n$. The theorem clearly holds for $m = 1$. Hence we can assume that $m \geq 2$ and n is minimal.

We first observe that by the definition of ω_{CP}, the order in which the tasks are executed corresponds precisely to using the list $L = (T_1, T_2, \ldots, T_n)$. Suppose in the corresponding timing diagram D_L, there is a task T_r with $r < n$ and $f_r = \omega_{\mathrm{CP}}$. If we consider the subset $\mathcal{T}' = \{T_1, \ldots, T_r\}$ with the list $L' = (T_1, \ldots, T_r)$, we see that the execution time ω' for $\mathcal{T}'$ using L' is just ω_{CP}. On the other hand, the optimal value ω_0' for $\mathcal{T}'$ certainly satisfies $\omega_0' \leq \omega_0$. Hence we have

$$\frac{\omega'}{\omega_0'} \geq \frac{\omega_{\mathrm{CP}}}{\omega_0} > \frac{4}{3} - \frac{1}{3m}$$

so that $\mathcal{T}'$ forms a smaller counterexample to the theorem. However, this contradicts the minimality of n. Thus we can assume that $f_k < \omega_{\mathrm{CP}}$ for $k < n$.

It is clear that

$$\omega_0 \geq \frac{1}{m} \sum_{i=1}^{n} \tau_i \tag{8}$$

Also, it follows that

$$\sum_{i=1}^{n-1} \tau_i \geq m s_n, \qquad \omega_L = s_n + \tau_n \tag{9}$$

where s_n denotes the starting time of T_n, since no processor is idle before T_n starts being executed. Therefore, we can write

$$\begin{aligned}\frac{\omega_L}{\omega_0} &= \frac{s_n + \tau_n}{\omega_0} \leq \frac{\tau_n}{\omega_0} + \frac{1}{m\omega_0} \sum_{i=1}^{n-1} \tau_i \\ &= \frac{(m-1)\tau_n}{m\omega_0} + \frac{1}{m\omega_0} \sum_{i=1}^{n} \tau_i \leq \frac{(m-1)\tau_n}{m\omega_0} + 1\end{aligned}$$

Since (7) does not hold for $\mathcal{T}$ by hypothesis, we have

$$1+\frac{(m-1)\tau_n}{m\omega_0}\geq\frac{\omega_L}{\omega_0}>\frac{4}{3}-\frac{1}{3m}$$

$$\frac{(m-1)\tau_n}{m\omega_0}>\frac{1}{3}-\frac{1}{3m}=\frac{m-1}{3m} \tag{10}$$

$$\tau_n>\frac{\omega_0}{3}$$

Hence if (7) is false, in an optimal solution (with timing diagram D_0), no processor can execute more than two tasks.

Suppose the configuration shown in Fig. 5.20 occurs in D_0, where $\tau_i > \tau_j$, $\tau_{i'} > \tau_{j'}$. If we interchange $\tau_{i'}$ and $\tau_{j'}$ to form the configuration shown in Fig. 5.21, the (possibly) new finishing time ω' in D_0' certainly satisfies $\omega' \leq \omega_0$. Also, if the configuration in Fig. 5.22 occurs, where $\tau_i > \tau_j$, moving $T_{i'}$ from the processor executing T_i to the processor executing T_j cannot increase the finishing time. Let us call either of the two preceding operations a *type I* operation.

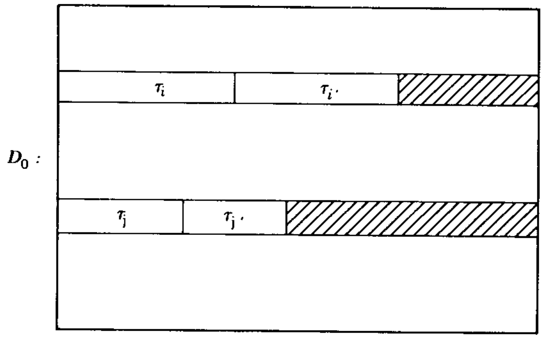


Figure 5.20 A portion of D_0.

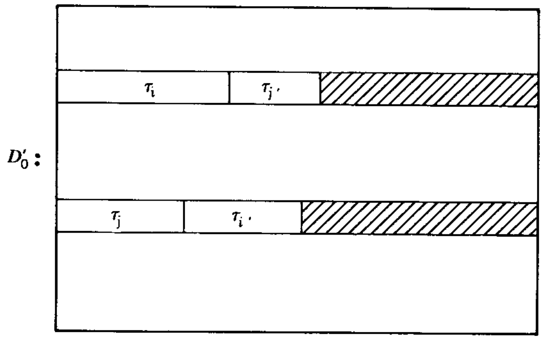


Figure 5.21 D_0 is modified.

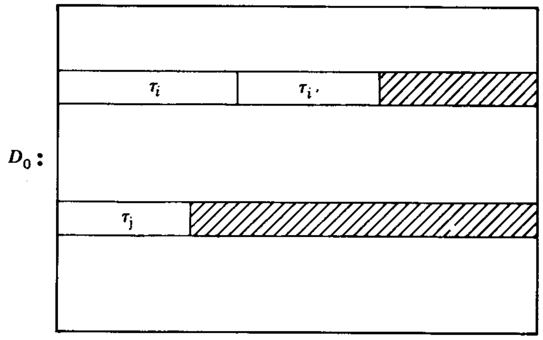


Figure 5.22 Another portion of D_0.

By a *type II* operation on D_0 we mean changing any occurrence of an "inversion" as illustrated in Fig. 5.23 to the "normalized" form in Fig. 5.24. Clearly, this operation does not affect ω_0.

For any timing diagram D we define a function $\mathcal{S}(D)$ as follows: if τ_i^* denotes the *least* time t such that for every time $t' \geq t$, the processor P_i is idle in D, then

$$\mathcal{S}(D) = \sum_{1 \leq i < j \leq m} |\tau_i^* - \tau_j^*|$$

It is not difficult to check:

1. If D' is obtained from D by a type I operation, $\mathcal{S}(D') < \mathcal{S}(D)$.
2. If D' is obtained from D by a type II operation, $\mathcal{S}(D') = \mathcal{S}(D)$.

Now we start from D_0 and apply all possible type I and type II operations until the resulting timing diagram D^* has no internal configurations to which either type of operation can be applied. That such a D^*

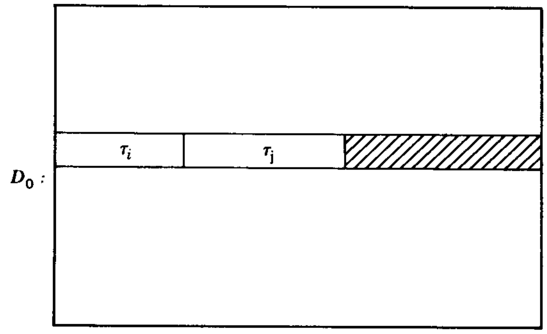


Figure 5.23 An inversion in D_0 with $\tau_i < \tau_j$.

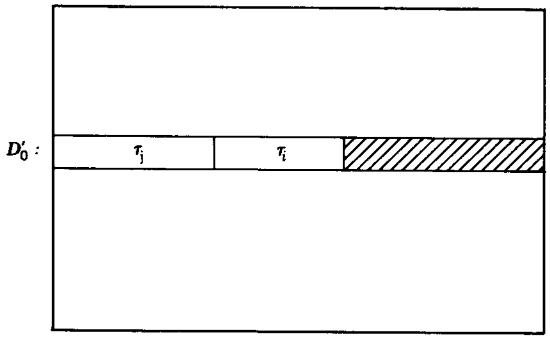


Figure 5.24 Normalized form.

exists follows because (*a*) there are only a finite number of possible arrangements of the *n* tasks on the *m* processors, (*b*) between any two type I operations only a finite number of type II operations can be performed, and (*c*) only a finite number of type I operations can be performed because of condition 1. Hence in D^* it follows that for any configuration of the form shown in Fig. 5.25, we have

$$\tau_i > \tau_j \quad \text{implies} \quad \tau_{j'} \geq \tau_{i'},\ \tau_i \leq \tau_k, \text{ and } \tau_i > \tau_{i'} \tag{11}$$

Thus by a suitable rearrangement of the processors of D^*, we can bring D^* into the form of Fig. 5.26, where

$$\tau_{k_1} \geq \tau_{k_2} \geq \cdots \geq \tau_{k_s}$$

$$\tau_{i_1} \geq \tau_{i_2} \geq \cdots \geq \tau_{i_t}$$

and, by (11),

$$\tau_{i_1'} \leq \tau_{i_2'} \leq \cdots \leq \tau_{i_t'}$$

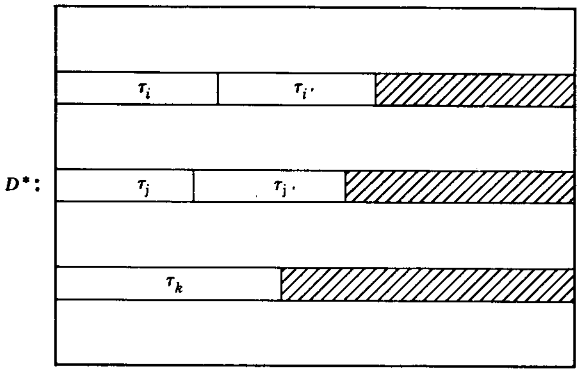


Figure 5.25 After operations are performed.

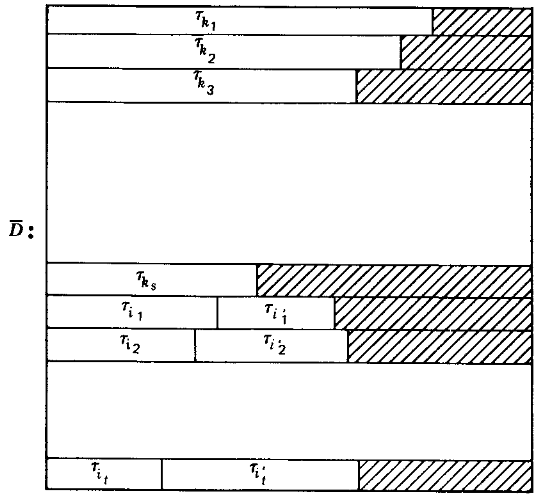


Figure 5.26 Further normalization of D^*.

But (11) also implies $\tau_{k_s} \geq \tau_{i_1}$ and $\tau_{i_t} \geq \tau_{i'_t}$. Hence combining these inequalities, we obtain

$$\tau_{k_1} \geq \cdots \geq \tau_{k_s} \geq \tau_{i_1} \geq \cdots \geq \tau_{i_t} \geq \tau_{i'_t} \geq \cdots \geq \tau_{i'_1} \tag{12}$$

Since none of the operations applied to D_0 causes an increase in ω_0, by the optimality of ω_0, the finishing time of $\bar{D}$ must also be ω_0. But $\bar{D}$ now looks very much like the timing diagram D_L obtained by using the decreasing-length list $L = (T_1, \ldots, T_n)$ (up to relabeling the tasks of equal length). In fact, the only way in which D_L could differ from $\bar{D}$ is in the assignment of the second-layer tasks $T_{i'_k}$. Specifically, a difference could occur only if for some pair T_{i_k}, $T_{i'_k}$, we have the situation represented in Fig. 5.27, where

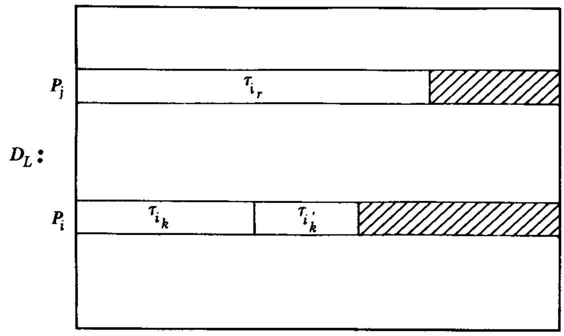


Figure 5.27 D_L differs from D_0.

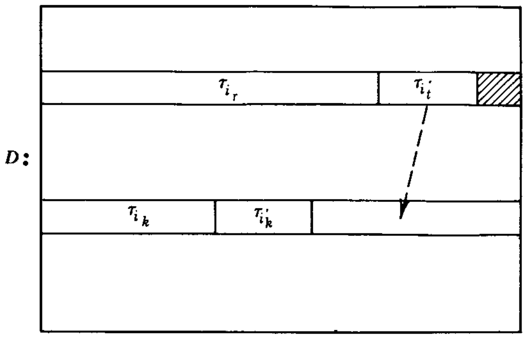


Figure 5.28 An unstable $\bar{D}$.

$\tau_{i_k} + \tau_{i_k'} \leq \tau_{i_r}$ for $r < k$. In this case, in D_L, $T_{i_r'}$ with length $\tau_{i_r'}$ might be assigned to P_j instead of P_i. However, if this situation were possible, we would have in $\bar{D}$ the situation illustrated in Fig. 5.28. Hence it would be possible to move $T_{i_r'}$ from P_i to P_j. Also, since the finishing time is not increased, it is still ω_0. But this is a *contradiction*, since we have an optimal solution in which a processor has three tasks assigned to it. Hence we conclude that D_L and $\bar{D}$ are isomorphic (in the obvious sense) and $\omega_0 = \omega_{CP}$. But this contradicts the hypothesis that $\omega_{CP}/\omega_0 > 4/3 - 1/(3m)$; hence (7) is proved.

To see that (7) is best possible, consider the following set of task lengths:

$$(\tau_1, \ldots, \tau_n) = (2m-1, 2m-1, 2m-2, 2m-2, \ldots, m+1, m+1, m, m, m)$$

where $n = 2m + 1$. That is,

$$\tau_k = 2m - \left\lfloor \frac{k+1}{2} \right\rfloor, \; 1 \leq k \leq 2m, \text{ and } \tau_{2m+1} = m$$

The corresponding timing diagrams for ω_{CP} and ω_0 appear in Fig. 5.29. As can be seen, we have

$$\frac{\omega_L}{\omega_0} = \frac{4}{3} - \frac{1}{3m}$$

as asserted. ▣

The following result verifies one's intuition in the following respect: as the proportion of the total execution time required by any one task tends to zero, the ratio ω/ω_0 tends to 1.

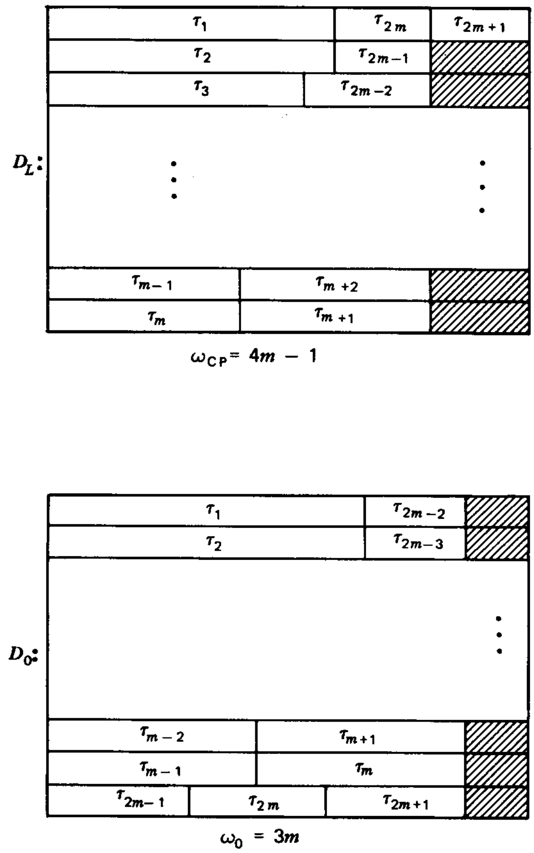


Figure 5.29 An extremal example.

Theorem 5.3 Suppose $<$ is empty. Then we have

$$\frac{\omega}{\omega_0} \le 1 + (m-1)\frac{\max \tau_i}{\sum_i \tau_i} \tag{13}$$

Proof Let τ^* denote $\max_i \tau_i$. Because of the rules under which the system operates, no processor is idle before time $\omega - \tau^*$. Since at least one processor is busy for ω units of time, we see that

$$\sum_i \tau_i \ge \omega + (m-1)(\omega - \tau^*)$$

Thus we have

$$\omega_0 \ge \frac{1}{m}(\omega + (m-1)(\omega - \tau^*))$$

that is,

$$\frac{\omega}{\omega_0} \le 1 + \frac{(m-1)}{m}\frac{\tau^*}{\omega_0} \le 1 + (m-1)\frac{\max \tau_i}{\sum_i \tau_i}$$

and the theorem is proved. ■

The next result helps quantify the tradeoff between the cost of computing "partially" optimal schedules and the corresponding decreases in the ratio ω/ω_0.

Theorem 5.4 [G2] Assume $<$ is empty and L_k is a list of the k tasks T_i having largest τ_i, $1 \le i \le k$, which is *optimal* for this set of k tasks. Form a list $L(k)$ by adjoining the remaining tasks arbitrarily and let $\omega(k)$ denote the finishing time using $L(k)$. Then we have

$$\frac{\omega(k)}{\omega_0} \le 1 + \frac{1 - \frac{1}{m}}{1 + \left\lfloor \frac{k}{m} \right\rfloor} \tag{14}$$

This is best possible if $k \equiv 0 (\mathrm{mod}\ m)$.

Proof If $\omega(k) = \omega_k$, the finishing time using $L(k)$, then $\omega(k) = \omega_0$ and the theorem holds. Therefore, we can assume $\omega(k) > \omega_k$. Also, we can assume $n > k$. Let τ^* denote $\max_{k+1 \le j \le n}\{\tau_j\}$. As in the proof of the preceding theorem, when using the list $L(k)$ no processor can be idle before time $\omega(k) - \tau^*$. Hence

$$\sum_{j=1}^{n} \tau_j \ge m(\omega(k) - \tau^*) + \tau^*$$

and

$$\omega_0 \ge \frac{1}{m}\sum_{j=1}^{n} \tau_j \ge \omega(k) - \left(\frac{m-1}{m}\right)\tau^*$$

There are at least $k + 1$ tasks having length $\ge \tau^*$. Thus some processor must execute at least $1 + \lfloor k/m \rfloor$ of these "long" tasks. This implies

$$\omega_0 \ge \left(1 + \left\lfloor \frac{k}{m} \right\rfloor\right)\tau^*$$

Combining the preceding inequalities, we obtain

$$\omega(k) \le \omega_0 + \left(\frac{m-1}{m}\right)\tau^* \le \omega_0\left(1 + \frac{1 - \frac{1}{m}}{1 + \left\lfloor \frac{k}{m} \right\rfloor}\right)$$

and (14) is proved.

To see that (14) is best possible when $k \equiv 0 (\bmod\, m)$, consider the following example.

Example 9 Define τ_i for $1 \leq i \leq k+1+m(m-1)$ by

$$\tau_i = \begin{cases} m & \text{for} \quad 1 \leq i \leq k+1 \\ 1 & \text{for} \quad k+2 \leq i \leq k+1+m(m-1) \end{cases}$$

For this set of task lengths and the list

$$L(k) = (T_1, \ldots, T_k, T_{k+2}, \ldots, T_{k+1+m(m-1)}, T_{k+1})$$

we have $\omega(k) = k + 2m - 1$. Since $\omega_0 = k + m$, then

$$\frac{\omega(k)}{\omega_0} = \frac{k+2m-1}{k+m} = 1 + \frac{1-1/m}{1+\lfloor k/m \rfloor} \qquad \text{since} \quad k \equiv 0 (\bmod\, m)$$

and the bound in (14) is achieved. This proves Theorem 5.4. ▣

For k sufficiently large, we will thus have a better bound on $\omega(k)$ than we have in (7) on ω_{CP}. However, finding an optimal list for the largest k tasks may itself be a hard problem when k is large. One might instead try to work from a *near*-optimal priority list for the largest k tasks.

If L'_k denotes a list formed from the k largest τ_i such that for some $\alpha \geq 0$ we have

$$\frac{\omega'_k}{\omega_0} \leq 1 + \alpha$$

and $L'(k)$ is formed from L'_k by adjoining the remaining tasks, the preceding arguments can be used to prove the following generalization of Theorem 5.4.

Theorem 5.4′ For L'_k defined as previously,

$$\frac{\omega'(k)}{\omega_0} \leq 1 + \max\left(\alpha, \frac{m-1}{m\lceil (k+1)/m \rceil}\right) \tag{15}$$

A more specific application of finding near optimal priority lists for the largest τ_i is given in [J3]. Here a guess ω'_0 is made as to the value of ω_0, and a partial schedule is constructed as follows: order the tasks by nonincreasing values of τ_i. Assign the first (largest) task to be the first task executed on the first processor. In general, assign the i^{th} task to be the next task executed on the lowest indexed processor to which it can be added without violating the deadline ω'_0. If it cannot be added to *any* processor without violating the deadline, halt. Let L_1 be the list of tasks assigned when the above procedure halts and L_2 the remaining portion of the list. Let L_1' be a permutation of L_1 which as a priority list will generate the same schedule as constructed above, and let L' be the list obtained from L_1' by appending L_2 to it. The following result has been proved.

Theorem [J3] For L' and ω_0' defined as above, $\omega_0' \geqslant \omega_0$ implies

$$\frac{\omega_{L'}}{\omega_0'} \leqslant \frac{5}{4}$$

This does not directly give us a bound on $\omega_{L'}/\omega_0$. However, the closer our guess ω_0' is to ω_0, the better the ratio $\omega_{L'}/\omega_0$ will be. Moreover, one can use repeated applications of the algorithm to *obtain* better guesses, using binary search techniques. If $L'(j)$ is the list obtained by the j^{th} iteration, we have the following result.

Theorem [J3]

$$\frac{\omega_{L'(j)}}{\omega_0} \leqslant \frac{5}{4}\left(1+\frac{1}{2^{j-1}}\right)$$

This bound will be better than that given in (7) for ω_{CP} if $m \geqslant 5$ and $j \geqslant 8$.

One might remark, however, that critical path scheduling for independent tasks (i.e. $<$ is empty) can itself obey bounds better than (7), if different measures of performance are used. In particular, a natural alternative that has been considered (see [CW]) is $\omega^*(L)$, which is defined for a list L by

$$\omega^*(L) = \sum_{i=1}^{n} \omega_i^2(L)$$

where $\omega_i(L)$ denotes the time at which processor P_i finishes using the list L. This performance measure has arisen in a study of partitioning information on secondary storage with the aim of minimizing access times. The following result has been proved.

Theorem [CW]

$$\frac{\omega_{CP}^*}{\omega_0^*} \leq \frac{25}{24}$$

On the other hand, examples exist [CW] for which

$$\frac{\omega_{CP}^*}{\omega_0^*} \geq \frac{37}{36} - \frac{1}{36m}$$

The complexity of the proof prohibits us from including it here.

In another secondary storage problem, a similar performance measure was considered; in particular, the function $(m-2)/2+(m/2)\omega^*$ was to be minimized. It has been shown that in this case [CC]

$$\frac{(m-2)/m + \omega_{CP}^*}{(m-2)/m + \omega_0^*} \leq 1 + \frac{1}{16(m-1)}$$

which again illustrates the increased effectiveness of CP assignments when the "second moment" measure is involved.

Finally, CP sequencing has also been specialized to the problem of obtaining reasonably "short" schedules that guarantee minimum mean flow time (see Chapter 3). In particular, suppose independent tasks are assigned as follows. The m smallest (rank 1) tasks are assigned first, one to a processor. The next m smallest (rank 2) tasks are then assigned, one to a processor and largest-task-first to the finishing times of the rank-1 tasks. This process continues with subsequent ranks until all tasks are assigned and each processor has a task from each rank (we assume that n is a multiple of m). With the largest-first (CP) criterion, one can show [CS] that

$$\frac{\omega'_{CP}}{\omega'_0} \leq \frac{5m-4}{4m-3}$$

is an optimal bound, where the prime signifies the restriction to schedules in the class of mean-flow-time-minimizing schedules.

5.3 REMARKS ON CRITICAL PATH SCHEDULING

As Example 7 shows, even if $<$ is a tree and all $\tau_i = 1$, it is still possible to have $\omega_L/\omega_0 = 2-(1/m)$ for a suitably bad list L. If critical path scheduling is used in this case, it is known (see Chapter 2) that $\omega_{CP} = \omega_0$. The following example, due to G. S. Graham [Gr], shows that if the τ_i are allowed to be arbitrary, even though $<$ is a tree, the ratio ω_{CP}/ω_0 can still be very close to 2.

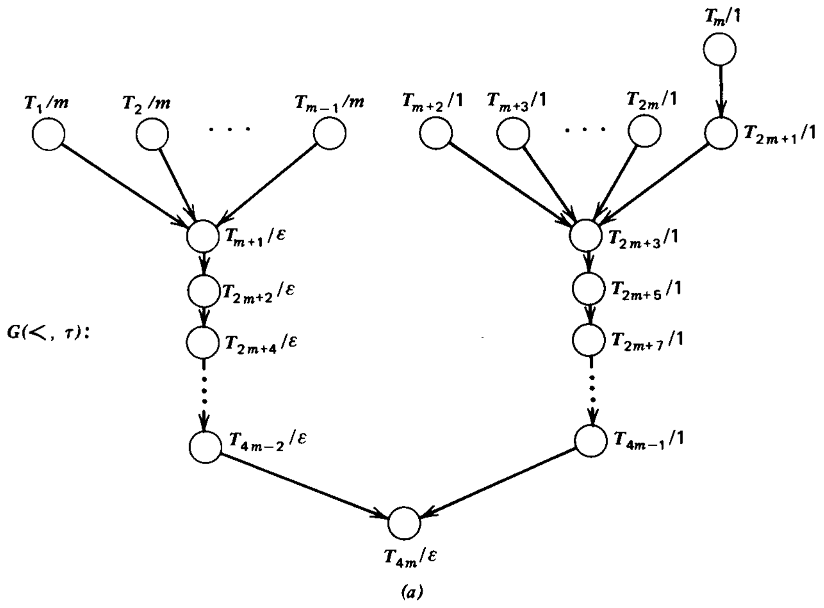
Example 9 Consider the tree and the critical path schedule illustrated in Figs. 5.30*a* and *b*, respectively. An optimal list can produce the schedule shown in Fig. 5.31. These diagrams show that

$$\frac{\omega_{CP}}{\omega_0} = \frac{2m+\epsilon}{(m+1)(1+\epsilon)}$$

which approaches $2-2/(m+1)$ as $\varepsilon \to 0$. Conceivably, this is the asymptotic worst-case behavior of ω_{CP} in the case that $<$ is a tree.

M. Kaufman [K2] has shown that if $<$ is a tree,

$$\frac{\omega_{CP}}{\omega_0} \leq 1 + (m-1)\frac{\max \tau_i}{\sum_i \tau_i}$$



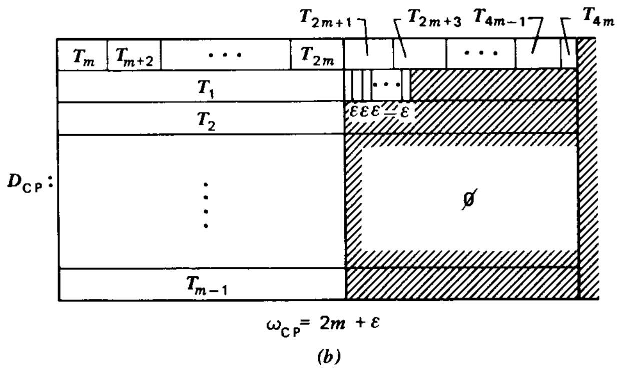


Figure 5.30 (*a*) Tree. (*b*) Critical path schedule.

which is analogous to the bound of Theorem 5.3 for $<$ empty and L arbitrary.

However, for general partial orders $<$ and arbitrary τ_i, critical path scheduling can result in the worst possible schedules, as the following example shows.

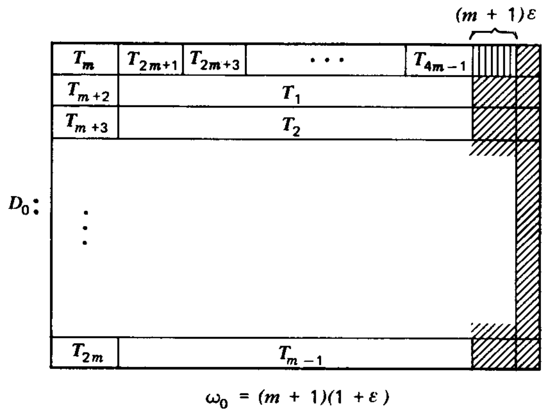


Figure 5.31 An optimal schedule.

Example 10 Suppose we have the graph shown in Fig. 5.32*a*. When executed by the critical path algorithm, we obtain the schedule in Fig. 5.32*b*. An optimal list is given by

$$L = (\varepsilon_1, \ldots, \varepsilon_m, U_1, \ldots, U_m, T_1, \ldots, T_m)$$

with the corresponding schedule given in Fig. 5.33. Evidently,

$$\frac{\omega_{\mathrm{CP}}}{\omega_0} = \frac{2m-1-2\varepsilon}{m} \to 2 - \frac{1}{m} \quad \text{as} \quad \varepsilon \to 0$$

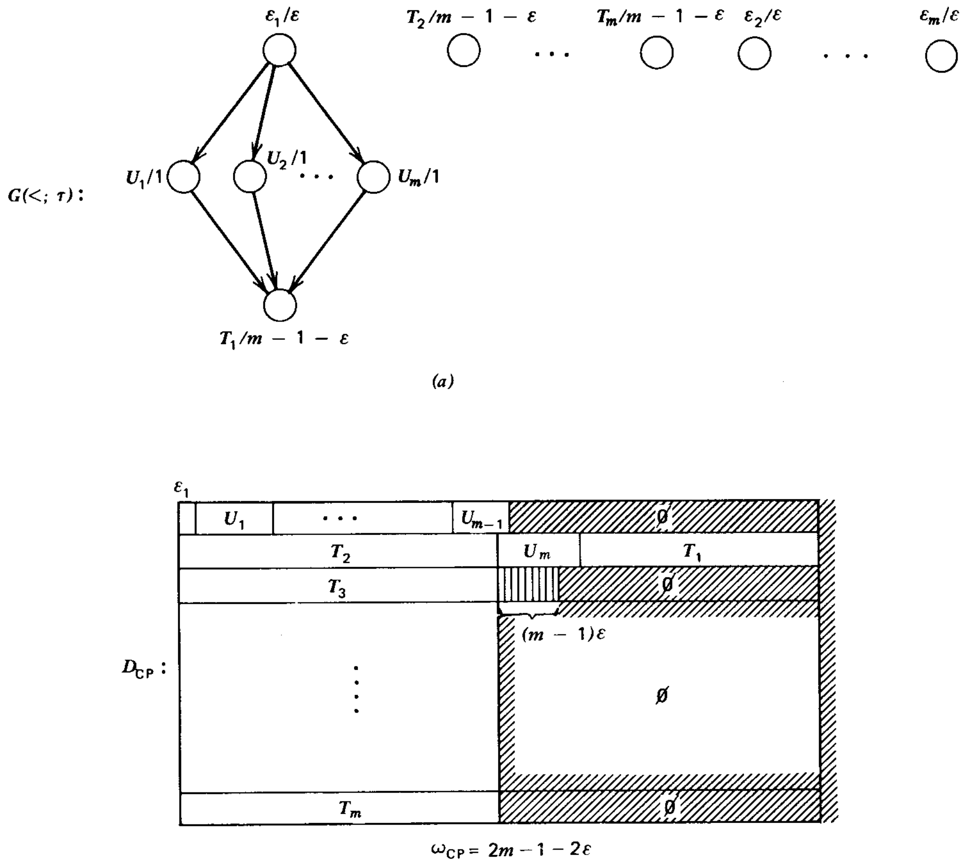
which, as we have seen, is as bad as the poorest performance any list can ever give. Note that this example also shows the algorithm that selects for the next task to execute that available task T which has the greatest sum $\Sigma_{T<T_i} \tau_i$ also can have a finishing time $\bar{\omega}$ with $\bar{\omega}/\omega_0$ arbitrarily close to $2-(1/m)$.

If we allow $<$ to be arbitrary but now restrict all the τ_i to be 1 then it is known [*CLi*] that the worst-case behavior of critical path scheduling is given by

$$\frac{\omega_{\mathrm{CP}}}{\omega_0} \le 2 - \frac{1}{m-1}, \quad m \ge 3 \tag{16}$$

Also examples can be given [*CLi*] which show that this bound is best possible. We give such an example for $m = 3$.

Example 11 Let all $\tau_i = 1$ for the graph shown in Fig. 5.34*a*. One possible list resulting from critical path scheduling (in which the worst possible choices are made for breaking ties) is $L = (T_1, T_2, T_3, \ldots, T_{12})$. The



$\varepsilon_1/\varepsilon$
$T_2/m - 1 - \varepsilon$
$T_m/m - 1 - \varepsilon$
$\varepsilon_2/\varepsilon$
$\varepsilon_m/\varepsilon$
$G(<; \tau)$:
$U_1/1$
$U_2/1$
$U_m/1$
$T_1/m - 1 - \varepsilon$
(a)
ε_1
U_1
U_{m-1}
$\emptyset$
T_2
U_m
T_1
T_3
$(m-1)\varepsilon$
D_{CP}:
T_m
$\omega_{CP} = 2m - 1 - 2\varepsilon$
(b)

Figure 5.32 A graph and critical path schedule.

corresponding schedule is shown in Fig. 5.34*b*. An optimal schedule is shown in Fig. 5.34*c* which corresponds to the list $L_0 = (T_4, T_8, T_1, T_2, T_3, T_5, T_6, T_7, T_9, T_{10}, T_{11}, T_{12})$. From the two schedules we find

$$\frac{\omega_{CP}}{\omega_0} = \frac{3}{2}$$

which is just the bound of (16) for $m = 3$.

Finally, we give a surprising example indicating that even when all $\tau_i = 1$, there may be *no* list L that is optimal for executing $\mathcal{T}$ with both $m = 2$ and $m = 3$.

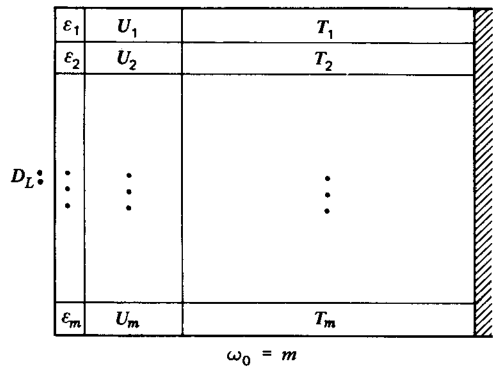


Figure 5.33 An optimal schedule.

Example 12 Consider the graph of Fig. 5.35, in which we suppose all τ_i to be 1. In any optimal list for $m = 2$, T_0 must precede *some* T_i, $1 \le i \le 6$. In any optimal list for $m = 3$, *all* T_i, $1 \le i \le 6$ must precede T_0!

5.4 SCHEDULING WITH MANY RESOURCES

We next examine the general case in which s, the number of resources, is nonzero. We recall that $\mathcal{R} = \{R_1, \ldots, R_s\}$ denotes the set of resources, $R_i(T_j)$ denotes the amount of resource R_i required by task T_j at all times during its execution, and for a fixed schedule, $r_i(t)$ denotes the total usage of resource R_i at time t. As might be expected, by allowing s to be nonzero, the ratio of ω/ω_0 can be much larger than before. We see this most clearly in the following result and example.

Theorem 5.5 [GG1] For $s = 1$, L, $<$, τ, m arbitrary, we have

$$\frac{\omega}{\omega_0} \le m \tag{17}$$

Proof The proof of (17) is immediate. We need only observe that

$$\omega \le \sum_{i=1}^{n} \tau_i \le m\omega_0$$

since at no time before time ω are all processors idle when using list L, and the number of processors busy at any time never exceeds m. ▣

To see that (17) is best possible, consider the following example.

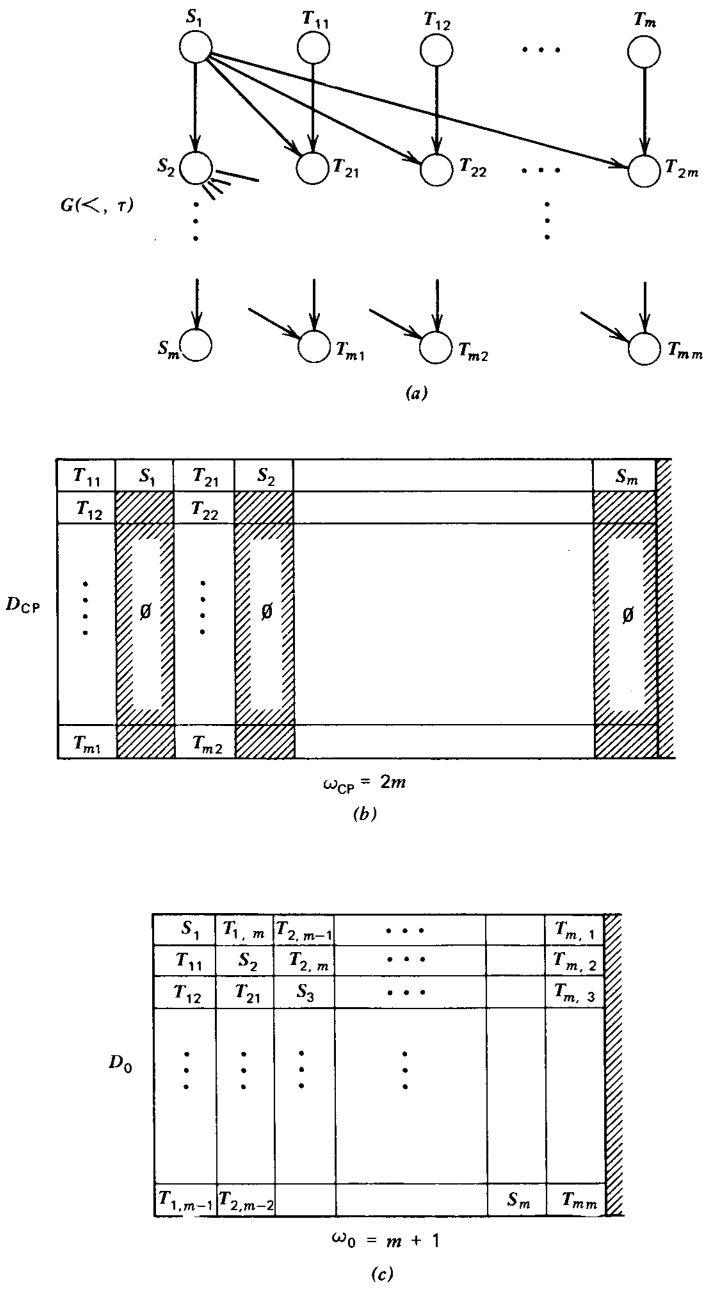


Figure 5.34 Conjectured worst case for CP scheduling.

$G(<, \tau)$:

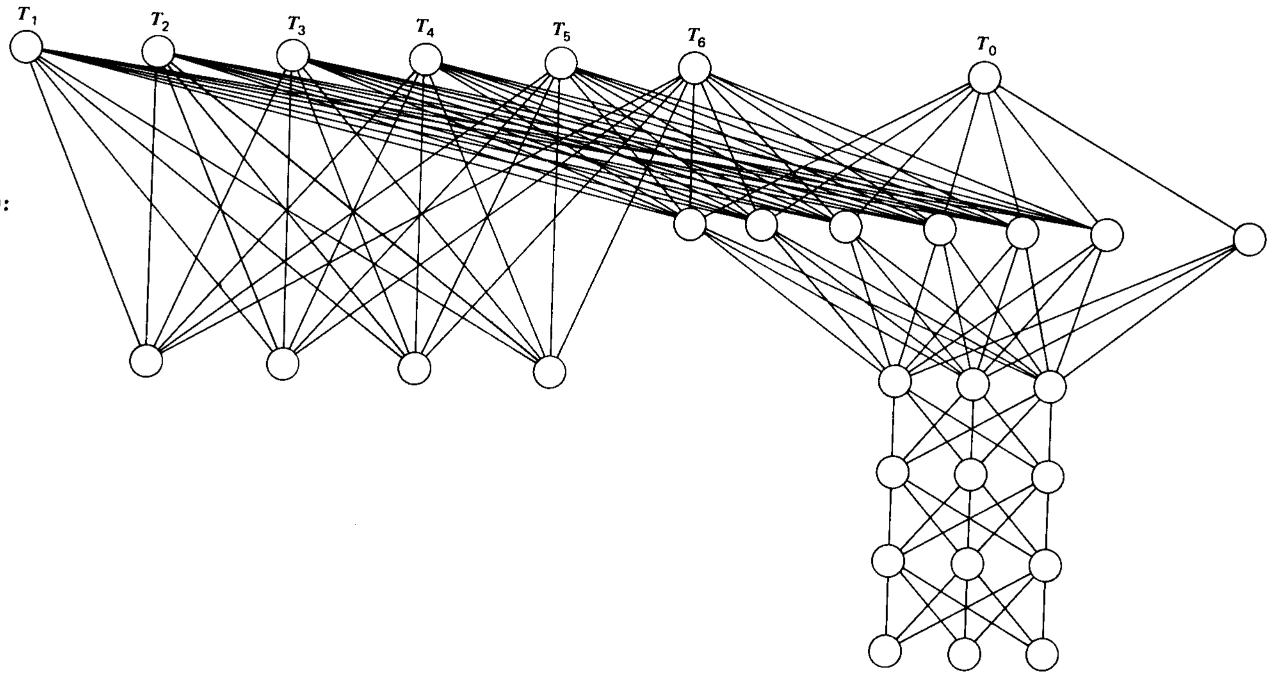


Figure 5.35 List scheduling example.

Example 13 Let $\mathscr{T} = \{T_1, \ldots, T_m, T'_1, \ldots, T'_m\}$, and let

$$s = 1,\ R_1(T_i) = \frac{1}{m},\ R(T'_i) = 1,\ 1 \le i \le m$$

$$\tau_i = 1, \quad \tau'_i = \varepsilon > 0$$

Let $<$ be defined by $T'_i < T_i$ for $1 \le i \le m$. For the lists

$$L = (T_1, \ldots, T_m, T'_1, \ldots, T'_m)$$
$$L' = (T'_1, \ldots, T'_m, T_1, \ldots, T_m)$$

we have, respectively, Figs. 5.36 and 5.37. Thus from the figures we have

$$\frac{\omega}{\omega'} = \frac{m + m\varepsilon}{1 + m\varepsilon} \to m \quad \text{as} \quad \varepsilon \to 0$$

A more interesting bound is given by the following theorem.

Theorem 5.6 [GG2] If $<$ is empty, $m \ge n$, s, L, τ arbitrary, then

$$\frac{\omega}{\omega_0} \le s + 1$$

Proof The proof requires several preliminary results.

Let G denote a graph with vertex set $V = V(G)$ and edge set $E = E(G)$. By a *valid labeling* λ of G, we mean a function $\lambda : V \to [0, \infty)$ satisfying

$$\lambda(a) + \lambda(b) \ge 1 \qquad \text{for all } e = \{a, b\} \in E \tag{18}$$

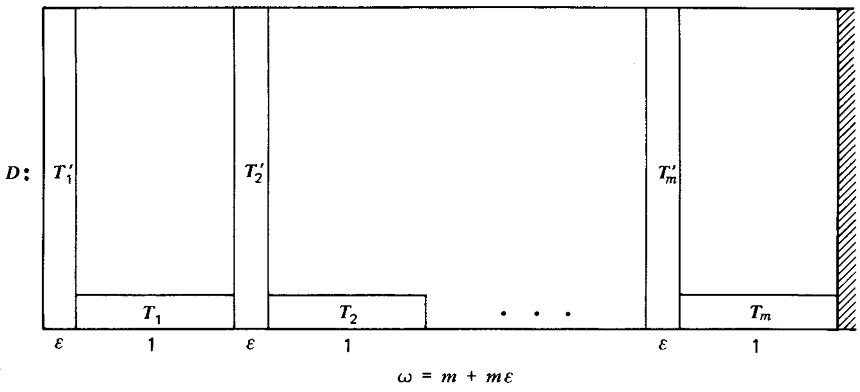


Figure 5.36 A bad schedule.

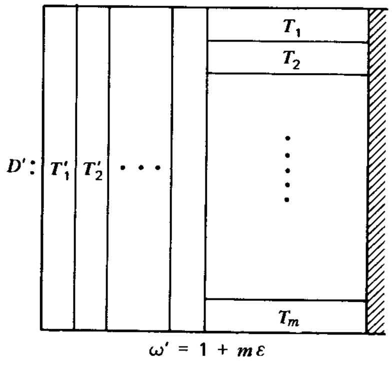


Figure 5.37 An optimal schedule.

Define the *score of* G, denoted by $\tau^*(G)$, by

$$\tau^*(G) = \inf_{\lambda} \sum_{v \in V} \lambda(v)$$

where the inf is taken over all valid labelings λ of G.

Lemma 5.1 For any graph G, there exists a valid labeling $\lambda : V \to \{0, \frac{1}{2}, 1\}$ such that

$$\tau^*(G) = \sum_{v \in V} \lambda(v)$$

Proof For the case that G is a bipartite graph (i.e., G has no odd cycles), a well-known theorem of König states that the number of edges in a maximum matching equals the cardinality of the minimum set of vertices of G incident to every edge of G. Thus for any bipartite graph G, there exists a valid labeling $\lambda : V \to \{0, 1\}$ such that

$$\tau^*(G) = \sum_{v \in V} \lambda(v)$$

For an arbitrary graph G, we construct a bipartite graph G_B as follows: for each vertex $v \in V(G)$ we form two vertices $v_1, v_2 \in V(G_B)$; for each edge $\{u, v\} \in E(G)$ we form two edges $\{u_1, v_2\}, \{u_2, v_1\} \in E(G_B)$. It is not difficult to verify that $\tau^*(G_B) = 2\tau^*(G)$ and furthermore, if $\lambda_B : V(G_B) \to \{0, 1\}$ is a valid labeling of G_B, then $\lambda : V(G) \to \{0, \frac{1}{2}, 1\}$ by $\lambda(v) = \frac{1}{2}(\lambda(v_1) + \lambda(v_2))$ is a valid labeling of G. ▣

For positive integers k and s, let $G(k, s)$ denote the graph with vertex set $\{0, 1, \ldots, (s + 1)k - 1\}$ and edge set consisting of all pairs $\{a, b\}$ for which $|a - b| \geq k$.

Lemma 5.2 Suppose $G(k, s)$ is partitioned into s spanning subgraphs H_i, $1 \le i \le s$. Then we have

$$\max_{1 \le i \le s} \{\tau^*(H_i)\} \ge k \tag{19}$$

Proof Assume the lemma is false i.e., there exists a partition of $G(k, s)$ into H_i, $1 \le i \le s$, such that $\tau^*(H_i) < k$ for $1 \le i \le s$. Thus by Lemma 5.1, for each i there exists a valid labeling $\lambda_i : V(H_i) \to \{0, \frac{1}{2}, 1\}$ such that

$$\sum_{v \in V(H_i)} \lambda_i(v) = \tau^*(H_i) < k \tag{20}$$

Let $A = \{a_1 < \cdots < a_p \mid \lambda_i(a_j) \le \frac{1}{2}$ for all i, $1 \le i \le s\}$ and let $\hat{\tau}$ denote $\Sigma_{i=1}^{s} \tau^*(H_i)$.

There are three cases.

1. $p \le k$. In this case we have

$$\hat{\tau} \ge k(s+1) - p \ge k(s+1) - k = ks$$

which contradicts (20).

2. $k < p < 2k + 1$. For each edge $\{a_j, a_{j+k}\}$, $1 \le j \le p - k$, there must exist an i such that $\lambda_i(a_j) + \lambda_i(a_{j+k}) \ge 1$. Thus

$$\hat{\tau} \ge k(s+1) - p + (p - k) = ks$$

again contradicting (20).

3. $p \ge 2k + 1$. We first note that for each vertex $v \in V(G(k, s))$, there exists an i such that $\lambda_i(v) \ge \frac{1}{2}$. For suppose $\lambda_i(v) = 0$ for $1 \le i \le s$. There must be some a_j such that $|a_j - v| \ge k$. But since $\lambda_i(a_j) \le \frac{1}{2}$ for all i, $\lambda_i(a_j) + \lambda_i(v) \le \frac{1}{2}$ for all i, which is a contradiction.

For each i, let n_i denote the number of vertices v such that $\lambda_i(v) = 1$. Then we have

$$|\{v \mid \lambda_i(v) > 0\}| \le 2k - 1 - n_i$$

since otherwise we would have

$$\sum_{v \in V(H_i)} \lambda_i(v) \ge n_i \cdot 1 + (2k - 2n_i) \cdot \frac{1}{2} = k$$

which contradicts (20). Therefore,

$$\sum_{i=1}^{s} |\{v \mid \lambda_i(v) > 0\}| \le (2k-1)s - \sum_{i=1}^{s} n_i \tag{21}$$

Let q denote the number of vertices v such that there is exactly one i for which $\lambda_i(v) > 0$. Then we have

$$\sum_{i=1}^{s} |\{v \mid \lambda_i(v) > 0\}| \ge 2(k(s+1) - q) + q \tag{22}$$

Combining (21) and (22) we obtain

$$q \geq 2k + s + \sum_{i=1}^{s} n_i \tag{23}$$

Of course we can assume without loss of generality that if $\lambda_i(v) = 1$, then $\lambda_j(v) = 0$ for all $j \neq i$. Hence by the definition of n_i, there must be at least $2k + s$ vertices, say, $b_1 < \cdots < b_{2k+s}$, such that

$$\sum_{i=1}^{s} \lambda_i(b_j) = \frac{1}{2}$$

that is, for each b_j there is a unique λ_i such that $\lambda_i(b_j) = \frac{1}{2}$ and $\lambda_l(b_j) = 0$ for all $l \neq i$. Thus if $|b_j - b_l| \geq k$, for some i, $\lambda_i(b_j) = \lambda_i(b_l) = \frac{1}{2}$. Since $|b_1 - b_{2k+s}| \geq k$, there exists i_0 such that $\lambda_{i_0}(b_1) = \lambda_{i_0}(b_{2k+s}) = \frac{1}{2}$. But by the same reasoning we must also have $\lambda_{i_0}(b_{k+j}) = \lambda_{i_0}(b_1) = \frac{1}{2}$ and $\lambda_{i_0}(b_{2k+s}) = \lambda_{i_0}(b_j) = \frac{1}{2}$ for $1 \leq j \leq k + s$. Therefore,

$$\tau^*(H_{i_0}) = \sum_{v \in V(H_{i_0})} \lambda_{i_0}(v) \geq (2k + s) \cdot \frac{1}{2} \geq k$$

which is a contradiction. This completes the proof of Lemma 5.2. ▣

Recall that when $\mathcal{T}$ is executed using a fixed list L, s_i denotes the time at which T_i starts to be executed. Because of the way the system is defined, each s_i is a sum of a subset of the τ_j's.

We can assume without loss of generality that $\omega_0 = 1$. Assume now that $\omega > s + 1$. Furthermore, suppose that for some k, each τ_i can be written as $\tau_i = l_i/k$, where l_i is a positive integer. Thus $l_i \leq k$, since $\tau_i \leq \omega_0 = 1$. Also, for $1 \leq i \leq s$, each $r_i(t)$ is constant on each interval $[l/m, (l+1)/m)$, this value being $r_i(l/m)$. It is important to note that since $<$ is empty and $m \geq n$, then for $t_1, t_2 \in [0, \omega)$ with $t_2 - t_1 \geq 1$, we must have

$$\max_{1 \leq i \leq s} \{r_i(t_1) + r_i(t_2)\} > 1$$

Otherwise, any task being executed at time t_2 should have been executed at time t_1 or sooner. Thus for each i, $1 \leq i \leq s$, we can define a graph H_i as follows:

$$V(H_i) = \{0, 1, \ldots, (s+1)k - 1\}$$

$\{a, b\}$ is an edge of H_i iff

$$r_i\left(\frac{a}{k}\right) + r_i\left(\frac{b}{k}\right) > 1 \tag{24}$$

Note that if $|a - b| \geq k$, then $\{a, b\}$ is an edge of at least one H_i. Hence it is not difficult to see that

$$G(k, s) \subseteq \bigcup_{1 \le i \le s} H_i$$

Note that by (24), the mapping $\lambda_i : V(H_i) \to [0, \infty)$ defined by $\lambda_i(a) = r_i(a/k)$ is a valid labeling of H_i. Since $G \subseteq G'$ implies $\tau^*(G) \le t^*(G')$ and the condition on the r_i in (24) is a strict inequality, it follows by Lemma 5.2 that

$$\max_i \left\{ \sum_{l=0}^{(s+1)k-1} r_i\left(\frac{l}{k}\right) \right\} = \max_i \left\{ \sum_{v \in V(H_i)} \lambda_i(v) \right\} > \max_i \{\tau^*(H_i)\} \ge k$$

But we must have

$$\frac{1}{k} \sum_{l=0}^{(s+1)k-1} r_i\left(\frac{l}{k}\right) \le \int_0^\infty r_i(t)\, dt \le 1, \quad 1 \le i \le s$$

that is,

$$\sum_{l=0}^{(s+1)k-1} r_i\left(\frac{l}{k}\right) \le k, \quad 1 \le i \le s$$

Since this is a *contradiction*, Theorem 5.6 is proved in the case that $\tau_i = l_i/k$ for positive integers k and l_i. Of course it follows immediately that Theorem 5.6 holds when all the τ_i are rational. The proof of the theorem can be completed by establishing the following lemma.

Lemma 5.3 Let $\tau = (\tau_1, \ldots, \tau_n)$ be a sequence of positive real numbers. Then for any $\varepsilon > 0$, there exists $\tau' = (\tau'_1, \ldots, \tau'_n)$ such that:

1. $|\tau'_i - \tau_i| < \varepsilon \quad$ for $\quad 1 \le i \le n$
2. For all $A, B \subseteq \{1, \ldots, n\}$,

$$\sum_{a \in A} \tau_a \le \sum_{b \in B} \tau_b \quad \text{iff} \quad \sum_{a \in A} \tau'_a \le \sum_{b \in B} \tau'_b$$

3. All τ'_i are positive rational numbers.

Remark The importance of condition 2 is that it guarantees that the order of execution of the T_i using the list L is the same for τ_i and τ'_i. Thus if L is used to execute $\mathcal{T}$, once using execution times τ_i and once using execution times τ'_i, the corresponding finishing times ω and ω' satisfy

$$|\omega - \omega'| \le n\varepsilon$$

Hence if there were an example $\mathcal{T}$ with $\omega/\omega_0 > s + 1$ and some of the τ_i irrational, we could construct another example $\mathcal{T}^*$ by slightly changing the τ_i to rational τ'_i, permitting the corresponding new finishing ω^* and $\omega^{*\prime}$ to satisfy

$$|\omega - \omega^*| \le n\varepsilon, \qquad |\omega' - \omega^{*\prime}| \le n\varepsilon$$

and, therefore, if ε is sufficiently small, we still have $\omega^*/\omega^{*\prime} > s + 1$. However, this would contradict what has already been proved.

Lemma 5.3 is implied by the following slightly more general result. The proof here is due to V. Chvatál [C].

Lemma 5.3′ Let $\mathscr{S}$ denote a finite system of inequalities of the form

$$\sum_{i=1}^{n} a_i x_i \ge a_0 \qquad \text{or} \qquad \sum_{i=1}^{n} a_i x_i > a_0$$

where the a_k are rational. If $\mathscr{S}$ has a real solution $(x_1, \ldots, x_n)$, then for any $\varepsilon > 0$, $\mathscr{S}$ has a *rational* solution $(x'_1, \ldots, x'_n)$ with $|x_i - x'_i| < \varepsilon$ for all i.

Proof We proceed by induction on n. For $n = 1$ the result is immediate. Now, let $\mathscr{S}$ be a system of inequalities in $n > 1$ variables, which is solvable in real numbers. $\mathscr{S}$ splits naturally into two classes: $\mathscr{S}_0$, the subset of inequalities involving x_n, and $\mathscr{S}_1$, the remaining inequalities of $\mathscr{S}$. Each inequality in $\mathscr{S}_1$ can be written in one of the following four ways:

a. $\alpha_0 + \sum_{i=1}^{n-1} \alpha_i x_i \le x_n$

b. $\alpha_0 + \sum_{i=1}^{n-1} \alpha_i x_i < x_n$

c. $\beta_0 + \sum_{i=1}^{n-1} \beta_i x_i \ge x_n$

d. $\beta_0 + \sum_{i=1}^{n-1} \beta_i x_i > x_n$

For each pair of inequalities, one of type a and one of type c, we consider the inequality

e. $\alpha_0 + \sum_{i=1}^{n-1} \alpha_i x_i \le \beta_0 + \sum_{i=1}^{n-1} \beta_i x_i$

Similarly, the pairs of types $\{a, d\}$, $\{b, c\}$, and $\{b, d\}$ give rise to inequalities

f. $\alpha_0 + \sum_{i=1}^{n-1} \alpha_i x_i < \beta_0 + \sum_{i=1}^{n-1} \beta_i x_i$

Let $\mathscr{S}^*$ be the set of all inequalities of types e and f we obtain from $\mathscr{S}_1$. Since by hypothesis, $\mathscr{S} = \mathscr{S}_0 \cup \mathscr{S}_1$ has a real solution $(x_1, \ldots, x_n)$, then

$\mathscr{S}_0 \cup \mathscr{S}^*$ has the real solution $(x_1, \ldots, x_{n-1})$. But $\mathscr{S}_0 \cup \mathscr{S}^*$ involves only $n-1$ variables, so that by the induction hypothesis, $\mathscr{S}_0 \cup \mathscr{S}^*$ has a rational solution $(x'_1, \ldots, x'_{n-1})$ with $|x_i - x'_i| < \varepsilon'$ for all i and any preassigned $\varepsilon' > 0$. Substituting the x'_i into a, b, c, and d, we obtain a set of inequalities

$$g.\ a' \le x_n, \quad b' < x_n, \quad c' \ge x_n, \quad d' > x_n$$

where a', b', c', and d' are rational. Since the x_i satisfy inequalities e and f, we have $a' \le c'$, $b' < c'$, $a' < d'$, $b' < d'$. Thus for any $\varepsilon > 0$, if ε' is chosen to be suitably small, there is a rational x'_n satisfying g and with $|x_n - x'_n| < \varepsilon$, completing the proof of Lemma 5.3'. This proves Lemma 5.3 and Theorem 5.6. ▣

The following example shows that the bound in the preceding theorem cannot be improved.

Example 14 Let $\mathscr{T} = \{T_1, T_2, \ldots, T_{s+1}, T'_1, T'_2, \ldots, T'_{sN}\}$, $m \ge s(N+1)+1 = n$, and suppose $<$ is empty. Define

$$\tau_i = 1, \qquad 1 \le i \le s+1, \qquad \tau'_i = \frac{1}{N}, \qquad 1 \le i \le sN$$

$$R_i(T_i) = 1 - \frac{1}{N}, \qquad R_i(T_j) = \frac{1}{Sn}, \qquad j \ne i, \qquad 1 \le i \le s$$

$$R_i(T'_j) = \frac{1}{N}, \qquad 1 \le j \le sN, \qquad 1 \le i \le s$$

$$\begin{aligned} L = (&T_1, T'_1, \ldots, T'_N, T_2, T'_{N+1}, \ldots, T'_{2N}, T_3, \ldots, \\ &T_{k+1}, T'_{kN+1}, T'_{kN+2}, \ldots, T'_{(k+1)N}, T_{k+2}, \ldots, T'_{sN}, T_{s+1}) \end{aligned}$$

$$L' = (T'_1, T'_2, \ldots, T'_{sN}, T_1, T_2, \ldots, T_{s+1})$$

It is easily checked that for this example

$$\omega = s+1, \qquad \omega' = 1 + \frac{s}{N}$$

Thus ω/ω' and consequently ω/ω_0 are both arbitrarily close to $s+1$ for N sufficiently large.

The last result in this section shows exactly the effect a processor constraint can have on the ratio ω/ω_0 (i.e., we do not assume $m \ge n$).

Theorem 5.7 [GG2] For $<$ empty, $m \ge 2$, s, L, τ arbitrary, we have

$$\frac{\omega}{\omega_0} \le \min\left\{\frac{m+1}{2},\ s + 2 - \frac{2s+1}{m}\right\} \tag{25}$$

Proof The proof consists of two main lemmas, each of which gives a bound on ω/ω_0 that is best possible for certain values of s and m. If X is a finite union of disjoint intervals in $[0, \omega)$, we let $\mu(X)$ denote the sum of the lengths of these intervals.

Lemma 5.4 If $<$ is empty and s, L, τ, m arbitrary, we have

$$\frac{\omega}{\omega_0} \leq \frac{m+1}{2}$$

Proof Let $I = \{t \mid |f(t)| = 1\}$, where we recall that $f(t)$ is defined to be that subset of tasks T_i which are being executed at time t (where we have a fixed list L under consideration). We first show

$$\mu(I) \leq \omega_0 \tag{26}$$

Consider the set T of tasks defined by

$$T = \bigcup_{t \in I} f(t)$$

For any pair of tasks T_i, T_j, belonging to T, there must exist some k, $1 \leq k \leq s$, such that

$$R_k(T_i) + R_k(T_j) > 1$$

since otherwise, one of those tasks should have been started earlier (unless $m = 1$, in which case the lemma is trivial). But this implies that in the optimal schedule no two members of T can be executed simultaneously. Therefore we have

$$\omega_0 \geq \sum_{T_i \in T} \tau_i \geq \mu(I)$$

which proves (26).

To complete the proof of Lemma 5.4, observe that at least two processors must be active at all times $t \in \bar{I} = [0, \omega) - I$. Thus,

$$\begin{aligned} m\omega_0 \geq \sum_{i=1}^{n} \tau_i &\geq 2\mu(\bar{I}) + \mu(I) \\ &= 2\omega - \mu(I) \\ &\geq 2\omega - \omega_0 \end{aligned}$$

and so

$$(m+1)\omega_0 \geq 2\omega \quad \blacksquare$$

Lemma 5.5 If $<$ is empty, $m \geq 3$ and s, L, τ arbitrary, then

$$\frac{\omega}{\omega_0} \leq s + 2 - \frac{2s+1}{m} \tag{27}$$

Proof Suppose we have a counterexample to the lemma. By Lemma 5.3 we can assume all the τ_i are rational; that is, there exists a positive integer k such that for each i, $1 \le i \le n$, there exists an integer l_i satisfying $\tau_i = l_i/k$. Without loss of generality we can also assume that $\omega_0 = 1$. Thus $\omega > s + 2 - (2s+1)/m$ and each l_i satisfies $1 \le l_i \le k$.

Consider the operation of the system using the list L. As before, let $I = \{t \in [0, \omega) \mid |f(t)| = 1\}$, $I' = \{t \in [0, \omega) \mid |f(t)| = n\}$ and let $\bar{I} = [0, \omega) - I$. By the proof of Lemma 5.4, $\mu(I) \le 1$. Since at least two processors are active at each time $t \in \bar{I}$,

$$\begin{aligned} m \ge \sum_{i=1}^{n} \tau_i &\ge m \cdot \mu(I') + \mu(I) + 2(\omega - \mu(I) - \mu(I')) \\ &\ge (m-2)\mu(I') + 2\omega - 1 \end{aligned}$$

or

$$\mu(I') \le \frac{m + 1 - 2\omega}{m - 2} \tag{28}$$

Since $\omega > s + 2 - (2s+1)/m$, we have

$$\begin{aligned} \mu(\bar{I}) &= \omega - \mu(I') \\ &\ge \omega - \frac{m+1-2\omega}{m-2} \\ &> s + 2 - \frac{2s+1}{m} - \frac{m + 1 - 2\left(s + 2 - \frac{2s+1}{m}\right)}{m-2} \\ &= s + 1 \end{aligned} \tag{29}$$

Now, observe that for any $t_1, t_2 \in \bar{I}$ satisfying $t_2 - t_1 \ge 1$, there must exist an i, $1 \le i \le s$, such that

$$r_i(t_1) + r_i(t_2) > 1 \tag{30}$$

Otherwise, some task being executed at time t_2 should have been started at time t_1 or sooner. Recalling that $\bar{I}$ is a collection of intervals, each having the form $[l/k, (l+1)/k)$ for some integer l, let $a_0 < a_1 < \cdots < a_p$ be integers such that

$$\bar{I} = \left\{\left[\frac{a_i}{k}, \frac{a_i+1}{k}\right) : 0 \le i \le p\right\}$$

Notice that (29) implies that $p \ge (s+1)k$. For each i, $1 \le i \le s$, we construct a graph H_i as follows:

$$V(H_i) = \{0, 1, 2, \ldots, (s+1)k - 1\}$$

$\{u, v\}$ is an edge of H_i iff

$$r_i\left(\frac{a_u}{k}\right) + r_i\left(\frac{a_v}{k}\right) > 1$$

Note that $|u - v| \geq k$ implies $|a_u - a_v| \geq k$, which, by (30), implies that $\{u, v\}$ is an edge of at least one H_i, $1 \leq i \leq s$. Hence it is not difficult to see that $G(k, s) \subseteq \bigcup_i H_i$. The same reasoning used in the proof of Theorem 5.6 can now serve to show that for some i, $1 \leq i \leq s$,

$$\int_0^\omega r_i(t)\, dt > 1$$

which contradicts the assumption that $\omega_0 = 1$. This completes the proof of Lemma 5.5. Theorem 5.7 follows by combining Lemmas 5.4 and 5.5. ▣

We now give examples to show that the bound given in the preceding theorem is best possible. These examples are slightly more complicated than those previously presented. We leave the verification of the asserted values of ω and ω' to the reader.

Example 15 We consider three cases.

(*i*) $2 \leq m \leq s + 1$, where $\mathcal{T} = \{T_0, T_1, T_2, \ldots, T_{m-1}, T'_1, T'_2, \ldots, T'_{m-1}\}$, $<$ is empty, s arbitrary, and

$$\tau_0 = 1, \qquad \tau_j = \tau'_j = \frac{1}{2}, \qquad 1 \leq j \leq m - 1$$

$$R_i(T_0) = \frac{1}{2m}, \qquad 1 \leq i \leq s$$

$$R_i(T_i) = R_i(T'_i) = \frac{1}{2}, \qquad 1 \leq i \leq m - 1$$

$$R_i(T_j) = R_i(T'_j) = \frac{1}{2m}, \qquad i \neq j, \qquad 1 \leq i \leq s, \qquad 1 \leq j \leq m - 1$$

$$L = (T_1, T'_1, T_2, T'_2, \ldots, T_{m-1}, T'_{m-1}, T_0)$$

$$L' = (T_0, T_1, T_2, \ldots, T_{m-1}, T'_1, T'_2, \ldots, T'_{m-1})$$

Then

$$\omega = \frac{m+1}{2}, \qquad \omega_0 = \omega' = 1$$

(*ii*) $s + 1 < m \geq 2s + 1$. For a suitably small $\varepsilon > 0$ and an arbitrary positive integer k, define $\varepsilon_i = \varepsilon(m - 1)^{i-2k}$, $1 \leq i \leq 2k$.

$$\mathcal{T} = \{T_0\} \cup \{T_{i,j} | 1 \leq i \leq m - 1, 1 \leq j \leq k\} \cup \{T'_{i,j} | 1 \leq i \leq m - 1, 1 \leq j \leq k\}$$

Also, $<$ is empty, s arbitrary, and

$$\tau_0 = 2k, \ \tau_{i,j} = \tau'_{i,j} = 1, \qquad 1 \le i \le m-1, \ 1 \le j \le k$$

$$R_i(T_0) = \varepsilon_1, \ 1 \le i \le s$$
$$R_i(T_{i,j}) = 1-(m-1)\varepsilon_{2j-1}, \qquad 1 \le i \le s, \ 1 \le j \le k$$
$$R_l(T_{i,j}) = \varepsilon_{2j-1}, \qquad l \ne i, \ 1 \le l \le s, \ 1 \le i \le m-1, \ 1 \le j \le k$$
$$R_i(T'_{s+i,j}) = 1-(m-1)\varepsilon_{2j}, \qquad 1 \le i \le m-s-1, \ 1 \le j \le k$$
$$R_l(T'_{i,j}) = \varepsilon_{2j}, \qquad l \ne i-s, \ 1 \le l \le s, \ 1 \le i \le m-1, \ 1 \le j \le k$$
$$L = (A_1, A_2, \dots, A_k, A'_1, A'_2, \dots, A'_{k-1}, A_0)$$

where

$$A_i = (B_{1,i}, B_{2,i}, \dots, B_{s,i}), \qquad 1 \le i \le k$$
$$B_{j,i} = (T_{j,i}, T'_{j,i}), \qquad 1 \le i \le k, \ 1 \le j \le s$$
$$A'_i = (B'_{1,i}, B'_{2,i}, \dots, B'_{m-1,i}), \qquad 1 \le i \le k-1$$
$$B'_{j,i} = (T'_{s+j,i}, T_{s+j,i+1}), \qquad 1 \le i \le k-1, \ 1 \le j \le m-1$$
$$A_0 = (T_0, T_{s+1,1}, T_{s+2,1}, \dots, T_{m-1,1}, T'_{s+1,k}, T'_{s+2,k}, \dots, T'_{m-1,k})$$
$$L' = (C_0, C_1, C'_1, C_2, C'_2, \dots, C_k, C'_k)$$

where

$$C_0 = (T_0)$$
$$C_i = (T_{1,i}, T_{2,i}, \dots, T_{m-1,i}), \qquad 1 \le i \le k$$
$$C'_i = (T'_{1,i}, T'_{2,i}, \dots, T'_{m-i,i}), \qquad 1 \le i \le k$$

Then

$$\omega = k(m+1) - (m-s-1)$$
$$\omega' = \omega_0 = 2k$$

and

$$\frac{\omega}{\omega_0} = \frac{m+1}{2} - \frac{(m-s-1)}{2k} \rightarrow \frac{m+1}{2} \qquad \text{as} \quad k \rightarrow \infty$$

(*iii*) $m > 2s+1$. For a suitably small $\varepsilon > 0$ and an arbitrary positive integer k', let $k = k'm$ and define $\varepsilon_i = \varepsilon(m-1)^{i-k}$, $1 \le i \le k$.

$$\mathcal{T} = \{T_0\} \cup \{T_{i,j} : 1 \le i \le m-1, \ 1 \le j \le k\}$$

$<$ is empty, s arbitrary

$$\tau_0 = k, \qquad \tau_{i,j} = 1, \qquad 1 \le i \le m-1,\ 1 \le j \le k$$
$$R_i(T_0) = \varepsilon_1, \qquad 1 \le i \le s$$
$$R_i(T_{i,j}) = 1-(m-1)\varepsilon_j, \qquad 1 \le i \le s,\ 1 \le j \le k$$
$$R_l(T_{i,j}) = \varepsilon_j, \qquad l \ne i,\ 1 \le l \le s,\ 1 \le i \le m-1,\ 1 \le j \le k$$
$$L = (A_1, A_2, \ldots, A_{m-2s-1}, B_1, B_2, \ldots, B_s, C)$$

where

$$A_i = (T_{2s+i,1}, T_{2s+i,2}, \ldots, T_{2s+i,k}), \qquad 1 \le i \le m-2s-1$$
$$B_i = (T_{i,1}, T_{s+i,2}, T_{i,2}, T_{s+i,3}, \ldots, T_{i,k-1}, T_{s+1,k}), \qquad 1 \le i \le s$$
$$C = (T_0, T_{s+1,1}, T_{s+2,1}, \ldots, T_{2s,1}, T_{1,k}, T_{2,k}, \ldots, T_{s,k})$$
$$L' = (T_0, D_1, D_2, \ldots, D_k)$$

where

$$D_i = (T_{1,i}, T_{2,i}, \ldots, T_{m-1,i}), \qquad 1 \le i \le k$$

Then we have

$$\omega = (s+2)k'm - (2s+1)k' - s$$
$$\omega' = \omega_0 = k'm$$

and

$$\frac{\omega}{\omega_0} = s+2-\frac{2s+1}{m}-\frac{s}{k'm} \to s+2-\frac{2s+1}{m} \qquad \text{as} \quad k' \to \infty \quad \blacksquare$$

5.5 BIN PACKING

In this section† we deal with the very important special case $s = 1$, $<$ is empty, all $\tau_i = 1$, and $m \ge n$. This has become known as the "bin-packing" problem in the literature for the obvious reason that the scheduling problem in this case is equivalent to the problem of "packing" the sequence of "weights" $(R_1(T_1), R_1(T_2), \ldots, R_1(T_n))$ into a minimum number of "bins" (i.e., unit time slots) of unit capacity (i.e., maximum resource usage is 1) so that no bin contains a total weight exceeding one.

As in Chapter 1 we note that the preceding is a complementary statement of the problem of minimizing the number of processors required to meet a given deadline ω common to all tasks. Without loss of generality we can assume that the execution times are normalized so that $\omega = 1$. In the latter problem the processors now become a "free" resource, and the processing time "resource" is bounded at $\omega = 1$. In the following it is convenient to take the former point of view, in which the problem is regarded as a special case of the model in the previous section.

† The contents and presentation of this section are based on portions of [JDUGG].

This problem is also equivalent to the one-dimensional stock-cutting problem (a variation of the knapsack problem), and efficient algorithms for obtaining optimal or near-optimal packings (i.e., schedules) have obvious practical applications—for example, in table formatting, file allocation, coil slitting (the formation of varying widths of coils of material formed from a single standard width), cable-length optimization problems, and generally, whenever a number of "pieces" of different "lengths" must be obtained from pieces having a standard length.

We modify the notation slightly for this section (to conform with that in standard use) as follows:

1. The "weight" $R_1(T_i)$ is denoted a_i. We can assume $0 < a_i \leq 1$.
2. The sequence of weights is denoted by $L = (a_1, a_2, \ldots, a_n)$.
3. The minimum number of bins into which the elements of L can be packed is denoted by L^*.
4. The ith bin, denoted by B_i, corresponds to the time interval $[i-1, i)$.

There are four bin-packing algorithms which will constitute our primary concern. In each of the algorithms, a_1 is packed first and a_k is packed before a_{k+1} is packed for all $k \geq 1$.

1. First-fit (or FF). Each a_k is placed into the lowest indexed bin into which it will fit.
2. Best-fit (or BF). Each a_k is placed into a bin for which the resulting unused capacity is minimal.
3. First-fit decreasing (or FFD). The L is arranged into a nonincreasing list and FF is applied to the resulting list.
4. Best-fit decreasing (or BFD). The same as 3, with BF replacing FF.

The corresponding numbers of bins required by these algorithms when used on a list L are denoted by FF(L), BF(L), FFD(L), and BFD(L), respectively.

We begin with a simple example illustrating a type of list for which FF and BF behave poorly.

Example 16 Let n be divisible by 18 and let δ satisfy $0 < \delta < 1/84$. Define a list $L = (a_1, a_2, \ldots, a_n)$ by

$$a_i = \begin{cases} \left(\frac{1}{6}\right) - 2\delta & \text{for} \quad 1 \leq i \leq \frac{n}{3} \\ \left(\frac{1}{3}\right) + \delta & \text{for} \quad \frac{n}{3} < i \leq \frac{2n}{3} \\ \left(\frac{1}{2}\right) + \delta & \text{for} \quad \frac{2n}{3} < i \leq n \end{cases}$$

Clearly, $L^* = n/3$, since the elements can be packed perfectly by placing one element of each type in each bin. However, it is easily checked that both the first fit and the best fit algorithm applied to L will result in a packing that consists of $n/18$ bins each containing six elements of size $(1/6) - 2\delta$, $n/6$ bins each containing two elements of size $(1/3) + \delta$, and $n/3$ bins each containing a single element of size $(1/2) + \delta$. The two packings are shown in Fig. 5.38. Thus we have

$$\frac{\mathrm{FF}(L)}{L^*} = \frac{\mathrm{BF}(L)}{L^*} = \frac{n/18 + n/6 + n/3}{n/3} = \frac{5}{3}$$

By slightly modifying the list L given in Example 16, we can force even worse behavior, as shown by the following result.

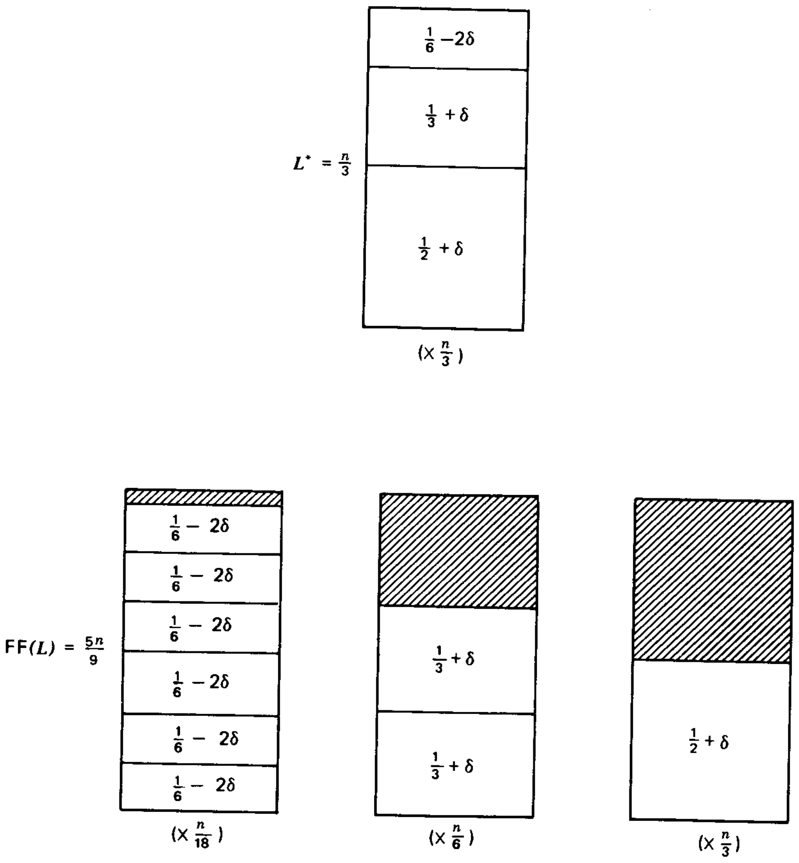


Figure 5.38 The 5/3 example.

Theorem 5.8 For every $k \geq 1$ there exists a list L with $L^* = k$ and

$$\mathrm{FF}(L) = \mathrm{BF}(L) > \frac{17}{10} L^* - 8$$

Proof As in the previous example, the elements of L belong to three regions, with sizes nearly equal to $\frac{1}{6}$, $\frac{1}{3}$, and $\frac{1}{2}$, respectively. The number of elements belonging to each region is the same. Those of the first region precede those of the second region, which in turn precede those of the third region in L.

Let n be a positive integer divisible by 17 and let δ be chosen so that $0 < \delta \ll 18^{-n/17}$. The first region consists of $n/17$ *blocks* of 10 numbers each. We denote the 10 numbers in the ith block of the first region by $a_{0,i}, a_{1,i}, \ldots, a_{9,i}$. These numbers are defined as follows, where $\delta_i = \delta \cdot 18^{(n/17)-i}$ for $1 \leq i \leq n/17$:

$$a_{0,i} = \left(\frac{1}{6}\right) + 33\delta_i$$

$$a_{1,i} = \left(\frac{1}{6}\right) - 3\delta_i$$

$$a_{2,i} = a_{3,i} = \left(\frac{1}{6}\right) - 7\delta_i$$

$$a_{4,i} = \left(\frac{1}{6}\right) - 13\delta_i$$

$$a_{5,i} = \left(\frac{1}{6}\right) + 9\delta_i$$

$$a_{6,i} = a_{7,i} = a_{8,i} = a_{9,i} = \left(\frac{1}{6}\right) - 2\delta_i$$

Let the first $10n/17$ elements in the list L be $(a_{0,1}, a_{1,1}, \ldots, a_{9,1}, a_{0,2}, a_{1,2}, \ldots, a_{9,2}, \ldots)$. Notice that $a_{0,i} + a_{1,i} + \cdots + a_{4,i} = (\frac{5}{6}) + 3\delta_i$ and $a_{5,i} + a_{6,i} + \cdots + a_{9,i} = (\frac{5}{6}) + \delta_i$. Thus for all i, the first five elements of block i will occupy bin $2i - 1$, and the last five elements of block i will occupy bin $2i$ when either the first-fit algorithm or the best-fit algorithm is applied. To see this, we need only observe that $a_{4,i}$, the smallest element in block i, will not fit into any of the preceding bins because the least filled of these, bin $2i - 2$, has contents totaling $(\frac{5}{6}) + \delta_{i-1} = (\frac{5}{6}) + 18\delta_i$. Also, the smallest of $a_{5,i}, a_{6,i}, \ldots, a_{9,i}$, which is $(\frac{1}{6}) - 2\delta_i$, will not fit into bin $2i - 1$, which has contents totaling $(\frac{5}{6}) + 3\delta_i$. Thus the $n/17$ blocks in the first region must occupy $2n/17$ bins.

We next turn to the second region. Here the elements are all about equal to $\frac{1}{3}$ and they are again divided into $n/17$ blocks of 10 elements each, the elements of the ith block being denoted by $b_{0,i}, b_{1,i}, \ldots, b_{9,i}$. In L, these elements all follow the $a_{j,i}$ and occur in the order ($b_{0,1}$, $b_{1,1}$, . . . , $b_{9,1}$, $b_{0,2}$, $b_{1,2}$, . . . , $b_{9,2}$, . . .). The values of the $b_{j,i}$ are defined as follows:

$$b_{0,i} = \left(\frac{1}{3}\right) + 46\delta_i$$

$$b_{1,i} = \left(\frac{1}{3}\right) - 34\delta_i$$

$$b_{2,i} = b_{3,i} = \left(\frac{1}{3}\right) + 6\delta_i$$

$$b_{4,i} = \left(\frac{1}{3}\right) + 12\delta_i$$

$$b_{5,i} = \left(\frac{1}{3}\right) - 10\delta_i$$

$$b_{6,i} = b_{7,i} = b_{8,i} = b_{9,i} = \left(\frac{1}{3}\right) + \delta_i$$

The elements of block i occupy bins $(2n/17) + 5i - 4$ through $(2n/17) + 5i$. These contain $b_{0,i}$ and $b_{1,i}$, $b_{2,i}$ and $b_{3,i}$, and so on. To see this, we observe that the contents of the five bins occupied by block i sum to, respectively,

$$\left(\frac{2}{3}\right) + 12\delta_i, \qquad \left(\frac{2}{3}\right) + 12\delta_i, \qquad \left(\frac{2}{3}\right) + 2\delta_i,$$

$$\left(\frac{2}{3}\right) + 2\delta_i, \qquad \text{and} \qquad \left(\frac{2}{3}\right) + 2\delta_i$$

Thus $b_{5,i} = (\frac{1}{3}) - 10\delta_i$ cannot fall into either of the first two bins and $b_{1,i} = (\frac{1}{3}) - 34\delta_i$ cannot fall into any of the bins of previous blocks, since these are all filled to at least level $(\frac{2}{3}) + 2\delta_{i-1} = (\frac{2}{3}) + 36\delta_i$. Thus the $n/17$ blocks in the second region occupy $5n/17$ bins.

The third region consists of $10n/17$ numbers, each equal to $(\frac{1}{2}) + \delta$. They clearly occupy one bin each. This completes the list L. The total number of bins required by applying either the first-fit algorithm or the best-fit algorithm is exactly n.

However, the elements of the list L can be packed into $(10n/17) + 1$ bins as follows. All but two of these bins contain one of the numbers $(\frac{1}{2}) + \delta$. The remaining space in each of these bins is filled with one of the following combinations:

1. $a_{j,i} + b_{j,i}$ for some i, j with $2 \leq j \leq 9$, $1 \leq i \leq n/17$
2. $a_{0,i} + b_{1,i}$ for some i, $1 \leq i \leq n/17$
3. $a_{1,i} + b_{0,i+1}$ for some i, $1 \leq i \leq n/17$

This leaves $b_{0,1}$, $a_{1,(n/17)}$ and one number $(\frac{1}{2}) + \delta$, which can easily be packed into the two remaining bins. We have therefore shown that $L^* \le 1 + 10n/17$ so that

$$\frac{\mathrm{FF}(L)}{L^*} \ge \frac{17n}{10n+17} > \frac{17}{10} - \frac{2}{L^*}$$

and similarly,

$$\frac{\mathrm{BF}(L)}{L^*} > \frac{17}{10} - \frac{2}{L^*}$$

To obtain values of L^* not congruent to 1 modulo 10, we can form the list L' by adjoining to L, k elements each with size 1, where k is a fixed integer ≤ 9. The preceding arguments then show

$$\mathrm{FF}(L') = \mathrm{FF}(L) + k \qquad \text{and} \qquad L'^* = L^* + k$$

so that

$$\frac{\mathrm{FF}(L')}{L'^*} \ge \frac{17}{10} - \frac{8}{L'^*}$$

with the same bound also holding for $\mathrm{BF}(L')/L'^*$. This proves Theorem 5.8. ▣

We now show that the examples constructed in the preceding proof are essentially the worst possible, that is, 17/10 is the asymptotic least upper bound of the ratios $\mathrm{FF}(L)/L^*$ and $\mathrm{BF}(L)/L^*$ for large L.

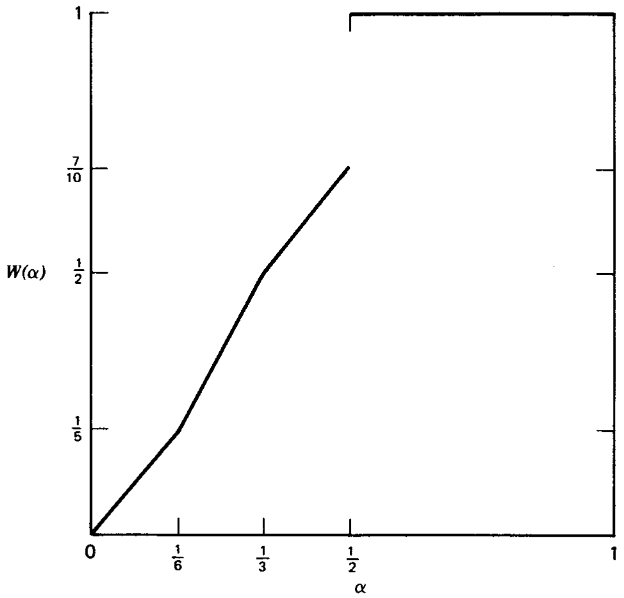


Figure 5.39 The function W.

Theorem 5.9 [U2], [GGU] For every list L, we have

$$\mathrm{FF}(L) \le \frac{17}{10} L^* + 2 \qquad \text{and} \qquad \mathrm{BF}(L) \le \frac{17}{10} L^* + 2$$

Proof We use only the two following properties of the FF and BF algorithms:

1. No element is placed in an empty bin unless it will not fit into *any* nonempty bin.
2. If there is a unique nonempty bin with lowest level, no element will be placed there unless it will not fit in any lower indexed bin. (The *level* of a bin refers simply to the sum of the weights it contains.)

Define the mapping $W: [0, 1] \to [0, 1]$ as follows (see Fig. 5.39):

$$W(\alpha) = \begin{cases} \frac{6}{5}\alpha & \text{for } 0 \le \alpha \le \frac{1}{6} \\ \frac{9}{5}\alpha - \frac{1}{10} & \text{for } \frac{1}{6} < \alpha \le \frac{1}{3} \\ \frac{6}{5}\alpha + \frac{1}{10} & \text{for } \frac{1}{3} < \alpha \le \frac{1}{2} \\ 1 & \text{for } \frac{1}{2} < \alpha \le 1 \end{cases}$$

Claim 1 Let some bin be filled with $b_1, b_2, \ldots, b_k$. Then

$$\sum_{i=1}^{k} W(b_i) \le \frac{17}{10}$$

Proof If $b \le \frac{1}{2}$, then $W(b)/b \le \frac{3}{2}$. The extreme ratio is reached only when $b = \frac{1}{3}$ and is less otherwise. Thus the claim is immediate unless one b_i is greater than $\frac{1}{2}$. We can take this one to be b_1, and we must now show that if

$$\sum_{i=2}^{k} b_i < \frac{1}{2}, \qquad \text{then} \quad \sum_{i=2}^{k} W(b_i) \le \frac{7}{10}$$

It should be noted that since the slope of $W(b)$ is the same in the regions $[0, \frac{1}{6}]$ and $[\frac{1}{3}, \frac{1}{2}]$, any b_i that is in the second region can be replaced without loss of generality by the two numbers $\frac{1}{3}$ and $b_i - \frac{1}{3}$. We therefore assume that $b_i \le \frac{1}{3}$ for $2 \le i \le k$. Moreover, if b_j and $b_{j'}$ are both $\le \frac{1}{6}$, they can be combined into one, and $\Sigma_i\, W(b_i)$ will not decrease; in fact it may increase. Thus we can assume that at most one of the b_i's, $i \ge 2$, is in the range $(0, \frac{1}{6}]$ and the rest are in $(\frac{1}{6}, \frac{1}{3}]$.

We have consequently reduced the proof to the consideration of four cases:

1. $k = 2$, $\quad b_2 \le \frac{1}{3}$
2. $k = 3$, $\quad \frac{1}{6} < b_2 \le b_3 \le \frac{1}{3}$
3. $k = 3$, $\quad b_2 \le \frac{1}{6} < b_3 \le \frac{1}{3}$
4. $k = 4$, $\quad b_2 \le \frac{1}{6} < b_3 \le b_4 \le \frac{1}{3}$

Case 1 is immediate because $b_2 \le \frac{1}{2}$ implies $W(b_2) \le \frac{7}{10}$. In case 2, $W(b_2) + W(b_3) = (\frac{9}{5})(b_2 + b_3) - \frac{1}{5} \le (\frac{9}{5})(\frac{1}{2}) - \frac{1}{5} = \frac{7}{10}$, since $b_2 + b_3 \le \frac{1}{2}$. For case 3, $W(b_2) + W(b_3) = (\frac{6}{5})b_2 + (\frac{9}{5})b_3 - \frac{1}{10} \leqslant \frac{1}{5} + \frac{3}{5} - \frac{1}{10} = \frac{7}{10}$. And finally, in case 4, $W(b_2) + W(b_3) + W(b_4) \leqslant (\frac{6}{5})b_2 + (\frac{9}{5})(b_3 + b_4) - \frac{1}{5} = (\frac{9}{5})(b_2 + b_3 + b_4) - (\frac{3}{5})b_2 - \frac{1}{5} \leqslant \frac{9}{10} - \frac{1}{5} = \frac{7}{10}$, since $b_2 + b_3 + b_4 \leqslant \frac{1}{2}$. ▣

Let us define the *coarseness* of a bin to be the largest α such that some bin with smaller index is filled to level $1 - \alpha$. The coarseness of the first bin is 0.

Claim 2 Suppose bins are filled according to either the FF or the BF algorithm, and some bin B has coarseness α. Then every member of B that was placed there *before* B was more than half full exceeds α.

Proof Until the bin has been filled to a level greater than $\frac{1}{2}$, it must be either empty, or the unique nonempty bin of lowest level (by property 1 of the placement algorithm), so by constraints 1 and 2 any element placed in the bin must not fit in any bin with lower index, hence must exceed α. ▣

Claim 3 Let a bin of coarseness $\alpha < \frac{1}{2}$ be filled with numbers $b_1 \ge b_2 \ge \cdots \ge b_k$ in the completed FF-packing (BF-packing). If $\Sigma_{i=1}^{k} b_i \ge 1 - \alpha$, then $\Sigma_{i=1}^{k} W(b_i) \ge 1$.

Proof If $b_1 > \frac{1}{2}$, the result is immediate, since $W(b_1) = 1$. We therefore assume that $b_1 \le \frac{1}{2}$. If $k \ge 2$, the second element placed in the bin was placed before the bin was more than half full; thus by claim 2 at least two of the elements exceed α. In particular, we must have $b_1 \ge b_2 > \alpha$. We consider several cases, depending on the range of α.

Case 1 $\alpha \le \frac{1}{6}$. Then $\Sigma_{i=1}^{k} b_i \ge 1 - \alpha \ge \frac{5}{6}$. Since $W(\beta)/\beta \ge \frac{6}{5}$ in the range $0 \le \beta \le \frac{1}{2}$, we immediately have $\Sigma_{i=1}^{k} W(b_i) \ge \frac{6}{5} \cdot \frac{5}{6} = 1$.

Case 2 $\frac{1}{6} \le \alpha \le \frac{1}{3}$. We consider subcases, depending on the value of k.

$k = 1$: Here, since $b_1 \le \frac{1}{2}$, we must have $1 - \alpha \le \frac{1}{2}$ or $\alpha \ge \frac{1}{2}$, which contradicts our assumption that $\alpha \le \frac{1}{3}$.

$k = 2$: If both b_1 and b_2 are $\ge \frac{1}{3}$, then $W(b_1) + W(b_2) \ge (\frac{6}{5} \cdot \frac{1}{3} + \frac{1}{10})2 = 1$. If both are $< \frac{1}{3}$, then $b_1 + b_2 < \frac{2}{3} < 1 - \alpha$, which contradicts our hypothesis. If $b_1 \ge \frac{1}{3}$, and $b_2 < \frac{1}{3}$, then since both must be greater than α, $\alpha < b_2 < \frac{1}{3} \le b_1 \le$

$\frac{1}{2}$. Hence $W(b_1)+W(b_2)=(\frac{9}{5})b_1-\frac{1}{10}+(\frac{6}{5})b_2+\frac{1}{10}=(\frac{6}{5})(b_1+b_2)+(\frac{3}{5})b_1$. Since $b_1+b_2\geq 1-\alpha$ and $b_1>\alpha$, we have $W(b_1)+W(b_2)\geq(\frac{6}{5})(1-\alpha)+(\frac{3}{5})\alpha=1+(\frac{1}{5}-(\frac{3}{5})\alpha)\geq 1$, since $\alpha\leq\frac{1}{3}$.

$k\geq 3$: As in the previous case, if two of the b_i are $\geq\frac{1}{3}$, the result is immediate. If $b_1\geq\frac{1}{3}>b_2\geq\alpha$, then

$$\begin{aligned}
&W(b_1)+W(b_2)+\sum_{i=3}^{k}W(b_i)\\
&\quad\geq\frac{6}{5}b_1+\frac{1}{10}+\frac{9}{5}b_2-\frac{1}{10}+\frac{6}{5}\sum_{i=3}^{k}b_i\\
&\quad=\frac{6}{5}\sum_{i=1}^{k}b_i+\left(\frac{3}{5}\right)b_2\geq\left(\frac{6}{5}\right)(1-\alpha)+\left(\frac{3}{5}\right)\alpha=1+\frac{1}{5}-\left(\frac{3}{5}\right)\alpha\geq 1
\end{aligned}$$

If $\frac{1}{3}>b_1\geq b_2>\alpha$, then

$$\begin{aligned}
W(b_1)+W(b_2)+\sum_{i=3}^{k}W(b_i)&\geq\left(\frac{9}{5}\right)(b_1+b_2)-\frac{1}{5}+\frac{6}{5}\sum_{i=3}^{k}b_i\\
&\geq\left(\frac{6}{5}\right)(1-\alpha)+\left(\frac{3}{5}\right)(2\alpha)-\frac{1}{5}\\
&=1+\left(\frac{6}{5}\right)\alpha-\left(\frac{6}{5}\right)\alpha=1
\end{aligned}$$

Case 3 $\frac{1}{3}<\alpha<\frac{1}{2}$. If $k=1$, we have $b_1\geq 1-\alpha>\frac{1}{2}$, so $W(b_1)=1$.
If $k\geq 2$, then $b_1\geq b_2>\frac{1}{3}$ and the result is immediate.

Claim 4 If a bin of coarseness $\alpha<\frac{1}{2}$ is filled with $b_1\geq\cdots\geq b_k$, and $\Sigma_{i=1}^{k}W(b_i)=1-\beta$, where $\beta>0$, then either

1. $k=1$ and $b_1\leq\frac{1}{2}$, or
2. $\Sigma_{i=1}^{k}b_i\leq 1-\alpha-(\frac{5}{9})\beta$

Proof If $k=1$ and $b_1>\frac{1}{2}$, it is impossible that $\beta>0$. Therefore, if condition 1 does not hold, we can assume that $k\geq 2$, hence $b_1\geq b_2\geq\alpha$, by the reasoning of the previous claim. Let $\Sigma_{i=1}^{k}b_i=1-\alpha-\gamma$. Then we can construct a bin filled with $b_3, b_4, \ldots, b_k$ and two other numbers δ_1 and δ_2, selected so that $\delta_1+\delta_2=b_1+b_2+\gamma$, $\delta_1\geq b_1$, $\delta_2\geq b_2$, and neither δ_1 nor δ_2 exceeds $\frac{1}{2}$. By the proof of claim 3 and because both δ_1 and δ_2 exceed α, we know that

$$\sum_{i=3}^{k}W(b_i)+W(\delta_1)+W(\delta_2)\geq 1$$

But since the slope W in range $[0,\frac{1}{2}]$ does not exceed $\frac{9}{5}$, it follows that $W(\delta_1)+W(\delta_2)\leq W(b_1)+W(b_2)+(\frac{9}{5})\gamma$. Therefore, $\gamma\geq(\frac{5}{9})\delta$, and condition 2 holds. ▣

We are now prepared to complete the proof of Theorem 5.9. Let $L=(a_1, a_2, \ldots, a_n)$ and $\bar{W}=\Sigma_{i=1}^{n}W(a_i)$. By claim 1, $(17/10)L^*\geq\bar{W}$.

Suppose that in the FF (BF) algorithm, bins $B'_1, B'_2, \ldots, B'_k$ are all the bins that receive at least one element and for which $\sum_j W(a_j) = 1 - \beta_i$ with $\beta_i > 0$, where j ranges over all elements in bin B'_i. We assume that $1 \le i < j \le k$ implies that B'_i had a smaller index than B'_j in the original indexing of all bins. Let γ_i be the coarseness of B'_i. Since B'_i contains no element exceeding $\frac{1}{2}$, we must have each $\gamma_i < \frac{1}{2}$. By claim 4 and the definition of coarseness,

$$\gamma_i \ge \gamma_{i-1} + \left(\frac{5}{9}\right)\beta_{i-1} \qquad \text{for} \quad 1 < i \le k$$

Thus

$$\sum_{i=1}^{k-1} \beta_i \le \frac{9}{5}\sum_{i=2}^{k}(\gamma_i - \gamma_{i-1}) = \frac{9}{5}(\gamma_k - \gamma_1) \le \frac{9}{5}\cdot\frac{1}{2} < 1$$

Since β_k cannot exceed 1, we have

$$\sum_{i=1}^{k} \beta_i \le 2$$

Applying claim 3, we obtain

$$\mathrm{FF}(L) \le \bar{W} + 2 \le (1.7)L^* + 2$$

and

$$\mathrm{BF}(L) \le \bar{W} + 2 \le (1.7)L^* + 2$$

completing the proof of the theorem. ▣

If the list $L = (a_1, \ldots, a_n)$ is such that for some $\alpha \le \frac{1}{2}$, all a_i are less than or equal to α, the worst-case behavior of the two placement algorithms is not as extreme. In particular, the following result holds.

Theorem 5.10 [GGU] For any positive $\alpha \le \frac{1}{2}$, let $k = \lfloor \alpha^{-1} \rfloor$. Then, we have:

1. For each $l \ge 1$, there exists a list $L = (a_1, \ldots, a_n)$ with all $a_i \in (0, \alpha]$ and $L^* = l$ such that

$$\mathrm{FF}(L) \ge \left(\frac{k+1}{k}\right)L^* - \frac{1}{k}$$

2. For any list $L = (a_1, \ldots, a_n)$ with all $a_i \in (0, \alpha]$,

$$\mathrm{FF}(L) \le \left(\frac{k+1}{k}\right)L^* + 2$$

Both 1 and 2 hold with FF replaced by BF.

Proof We first describe how to construct lists L, with no element exceeding α, for which

$$\frac{\mathrm{FF}(L)}{L^*}=\frac{\mathrm{BF}(L)}{L^*}=\frac{k+1}{k}-\frac{1}{kL^*}$$

Let l be any positive integer. The list L is composed of elements that are all very close to $1/(k+1)$. The elements are of two types, described as follows:

$$b_j=\frac{1}{k+1}-k^{2j+1}\delta, \qquad j=1,2,\ldots,l-1$$

$$a_{1j}=a_{2j}=\cdots=a_{kj}=\frac{1}{m+1}+k^{2j}\delta, \qquad j=1,2,\ldots,l$$

where $\delta>0$ is chosen suitably small. The list L has the a-type elements occurring in nonincreasing order and the b-type elements occurring in strictly increasing order, interspersed so that each successive pair b_j and b_{j-1} of b-type elements has precisely k a-type elements occurring in between. The list L is then completely specified by the property that b_{l-1} occurs as the second element. We leave it for the reader to verify that

$$\mathrm{FF}(L)=\mathrm{BF}(L)=\left\lceil\frac{l(k+1)-1}{k}\right\rceil$$

It is easy to see that the elements of L can be packed optimally by placing $b_j, a_{1j}, a_{2j}, \ldots, a_{kj}$ in a single bin for each $j=1,\ldots,l-1$ and placing $a_{1l}, a_{2l}, \ldots, a_{kl}$ in one additional bin. This gives $L^*=l$. We then have

$$\frac{\mathrm{FF}(L)}{L^*}=\frac{\mathrm{BF}(L)}{L^*}\geq\frac{l(k+1)-1}{kl}=\frac{k+1}{k}-\frac{1}{kL^*}$$

The upper bound is also easily proved. Suppose that the list L contains no element exceeding $1/k$, k an integer.

Consider an FF-packing of L. Every bin, except possibly the last bin, contains at least k elements. Disregarding the last bin, suppose two bins B_i and B_j, $i<j$, each contain elements totaling less than $k/(k+1)$. Then since B_j contains k elements, B_j must contain an element with size less than $1/(k+1)$. But this element would have fit in B_i and thus could not have been placed in B_j by FF, a contradiction. Thus all but at most two bins must contain elements totaling at least $k/(k+1)$. Thus, letting $w(L)$ denote the sum of all elements in L, we have

$$L^*\geq w(L)\geq\frac{k}{k+1}(\mathrm{FF}(L)-2)$$

so that

$$\mathrm{FF}(L) \le \frac{(k+1)L^*}{k} + 2$$

A similar but slightly more complicated argument can be used to prove this for BF. This proves the theorem. ▣

It follows at once from Theorem 5.10 that if $\alpha \le \frac{1}{2}$ and $L = (a_1, \ldots, a_n)$ with all $a_i \in (0, \alpha]$ then for all $\varepsilon > 0$

$$\frac{\mathrm{FF}(L)}{L^*} \le 1 + \lfloor \alpha^{-1} \rfloor^{-1} + \varepsilon$$

provided L^* is sufficiently large. The same bound also holds for BF.

The deepest results currently known concerning bin packing involve the first-fit decreasing (FFD) and best-fit decreasing (BFD) algorithms. To describe these results, we let $R_{\mathrm{FFD}}(\alpha)$ denote $\overline{\lim}_{L^* \to \infty} \mathrm{FFD}(L)/L^*$ where L ranges over all lists for which all elements a_i of L satisfy $a_i \in (0, \alpha]$. For example, the preceding results imply

$$R_{\mathrm{FF}}(\alpha) = \begin{cases} \frac{17}{10} & \text{for} \quad \alpha \in (\frac{1}{2}, 1] \\ 1 + \lfloor \alpha^{-1} \rfloor^{-1} & \text{for} \quad \alpha \in (0, \frac{1}{2}] \end{cases}$$

Theorem 5.11 [J1]

$$R_{\mathrm{FFD}}(\alpha) = \begin{cases} \dfrac{11}{9} & \text{for} \quad \alpha \in \left(\dfrac{1}{2}, 1\right] \\ \dfrac{71}{60} & \text{for} \quad \alpha \in \left(\dfrac{8}{29}, \dfrac{1}{2}\right] \\ \dfrac{7}{6} & \text{for} \quad \alpha \in \left(\dfrac{1}{4}, \dfrac{8}{29}\right] \\ \dfrac{23}{20} & \text{for} \quad \alpha \in \left(\dfrac{1}{5}, \dfrac{1}{4}\right] \end{cases}$$

The only known proof of Theorem 5.11, due to D. S. Johnson [J1], [JDUGG], is highly ingenious and rather complicated, exceeding 100 pages in length. Needless to say, space limitations prevent us from including it here. The difficult part of the proof—namely, establishing an upper bound on $R_{\mathrm{FFD}}(\alpha)$—is based on essentially the same strategy used in obtaining the upper bounds for FF and BF. That is, a "weighting function" is defined which assigns real numbers or "weights" to the elements of L, depending on their size, in such a way that

1. The total "weight" of all the elements in the list L differs from the total number of bins used in the particular packing under consideration (e.g., FF or FFD) by no more than some absolute constant c.

2. The total weight of any legally packed bin must be less than some fixed constant c'.

For FF we had $c' = 17/10$ and $c = 2$; for FFD one can choose $c' = 11/9$ and $c = 4$.

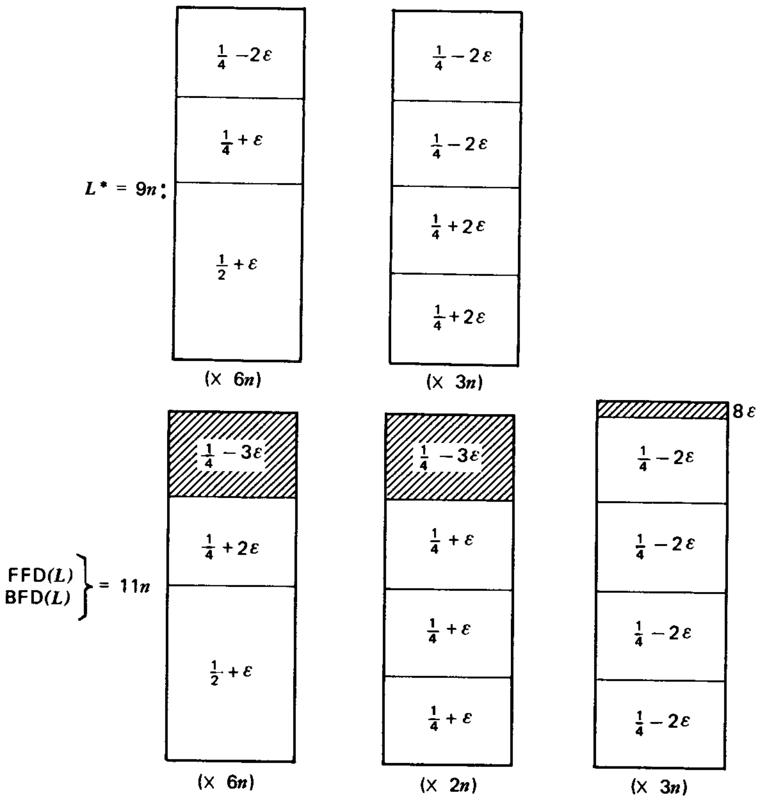
As in the case of FF and BF, the bounds of Theorem 5.11 for FFD also apply to BFD. However, there is a lack of symmetry between the FFD algorithm and the BFD algorithm, as indicated by the following result (which is also required in the proof that $R_{\text{BFD}}(1) = \frac{11}{9}$).

Theorem 5.12 [GGU], [JDUGG] For all lists $L = (a_1, \ldots, a_n)$, we have

1. $\text{BFD}(L) \leq \text{FFD}(L)$ if all $a_i \in [\frac{1}{6}, 1]$
2. $\text{BFD}(L) = \text{FFD}(L)$ if all $a_i \in [\frac{1}{5}, 1]$

Even the proof of Theorem 5.12 runs about 15 pages and is not given here.

Possibly, the complete form of Theorem 5.11 is given by the following conjecture [J2].



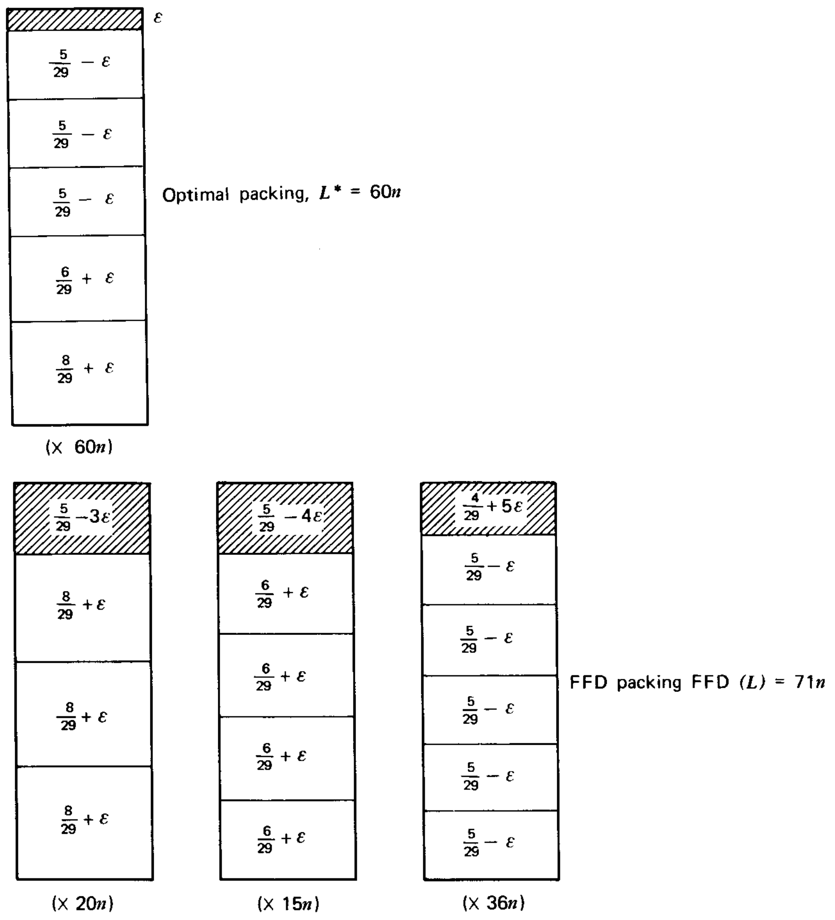
$L^* = 9n$:
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} + \varepsilon$
$\frac{1}{2} + \varepsilon$
(X 6n)
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} + 2\varepsilon$
$\frac{1}{4} + 2\varepsilon$
(X 3n)
FFD(L)
BFD(L)
$= 11n$
$\frac{1}{4} - 3\varepsilon$
$\frac{1}{4} + 2\varepsilon$
$\frac{1}{2} + \varepsilon$
(X 6n)
$\frac{1}{4} - 3\varepsilon$
$\frac{1}{4} + \varepsilon$
$\frac{1}{4} + \varepsilon$
$\frac{1}{4} + \varepsilon$
(X 2n)
8ε
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} - 2\varepsilon$
(X 3n)

Figure 5.40 An 11/9 example.

Conjecture

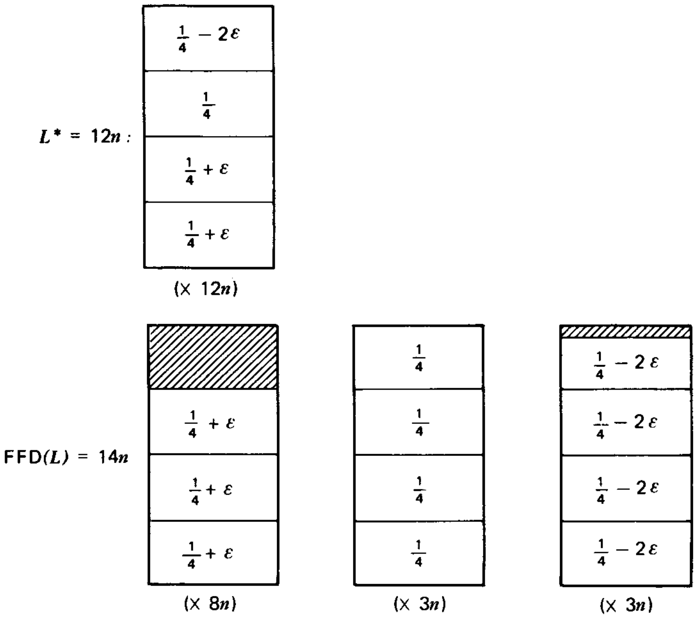
$$R_{\mathrm{FFD}}(\alpha) = \begin{cases} \dfrac{11}{9} & \text{for } \alpha \in \left(\dfrac{1}{2}, 1\right] \\ \dfrac{71}{60} & \text{for } \alpha \in \left(\dfrac{8}{29}, \dfrac{1}{2}\right] \\ \dfrac{7}{6} & \text{for } \alpha \in \left(\dfrac{1}{4}, \dfrac{8}{29}\right] \\ 1 + \dfrac{k-2}{k(k-1)} & \text{for } \alpha \leq \dfrac{1}{4}, \qquad k = \lfloor \alpha^{-1} \rfloor \end{cases}$$

These values of $R_{\mathrm{FFD}}(\alpha)$ for $\alpha \leq \frac{1}{4}$ are known to be upper bounds by straightforward extensions of the examples given in Figs. 5.42 and 5.43.



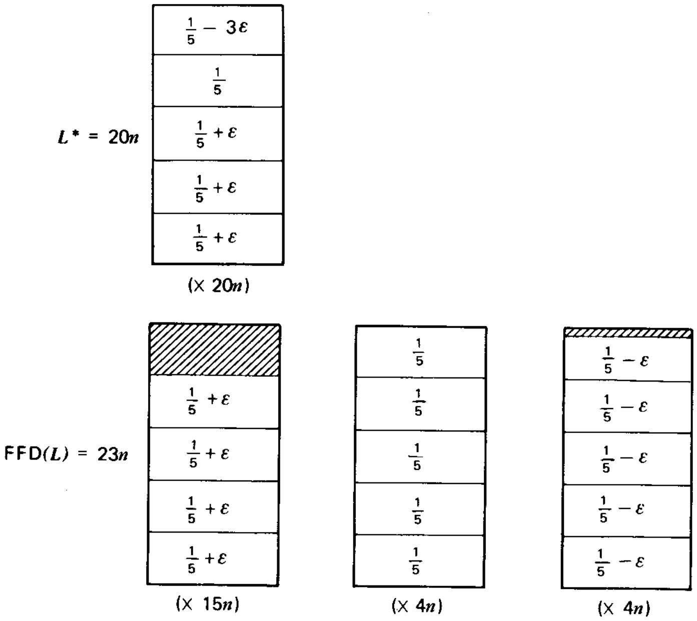
ε
$\frac{5}{29} - \varepsilon$
$\frac{5}{29} - \varepsilon$
$\frac{5}{29} - \varepsilon$
$\frac{6}{29} + \varepsilon$
$\frac{8}{29} + \varepsilon$
Optimal packing, $L^* = 60n$
(X 60n)
$\frac{5}{29} - 3\varepsilon$
$\frac{8}{29} + \varepsilon$
$\frac{8}{29} + \varepsilon$
$\frac{8}{29} + \varepsilon$
(X 20n)
$\frac{5}{29} - 4\varepsilon$
$\frac{6}{29} + \varepsilon$
$\frac{6}{29} + \varepsilon$
$\frac{6}{29} + \varepsilon$
$\frac{6}{29} + \varepsilon$
(X 15n)
$\frac{4}{29} + 5\varepsilon$
$\frac{5}{29} - \varepsilon$
$\frac{5}{29} - \varepsilon$
$\frac{5}{29} - \varepsilon$
$\frac{5}{29} - \varepsilon$
$\frac{5}{29} - \varepsilon$
(X 36n)
FFD packing FFD $(L) = 71n$

Figure 5.41 A 71/60 example.



$L^* = 12n$:
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4}$
$\frac{1}{4} + \varepsilon$
$\frac{1}{4} + \varepsilon$
(× $12n$)
FFD(L) = $14n$
$\frac{1}{4} + \varepsilon$
$\frac{1}{4} + \varepsilon$
$\frac{1}{4} + \varepsilon$
(× $8n$)
$\frac{1}{4}$
$\frac{1}{4}$
$\frac{1}{4}$
$\frac{1}{4}$
(× $3n$)
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} - 2\varepsilon$
$\frac{1}{4} - 2\varepsilon$
(× $3n$)

Figure 5.42 A 7/6 example.



$L^* = 20n$
$\frac{1}{5} - 3\varepsilon$
$\frac{1}{5}$
$\frac{1}{5} + \varepsilon$
$\frac{1}{5} + \varepsilon$
$\frac{1}{5} + \varepsilon$
(× $20n$)
FFD(L) = $23n$
$\frac{1}{5} + \varepsilon$
$\frac{1}{5} + \varepsilon$
$\frac{1}{5} + \varepsilon$
$\frac{1}{5} + \varepsilon$
(× $15n$)
$\frac{1}{5}$
$\frac{1}{5}$
$\frac{1}{5}$
$\frac{1}{5}$
$\frac{1}{5}$
(× $4n$)
$\frac{1}{5} - \varepsilon$
$\frac{1}{5} - \varepsilon$
$\frac{1}{5} - \varepsilon$
$\frac{1}{5} - \varepsilon$
$\frac{1}{5} - \varepsilon$
(× $4n$)

Figure 5.43 A 23/20 example.

In Figs. 5.40 through 5.45 are presented self-explanatory examples showing that the various ranges over which the a_i are allowed to vary in Theorems 5.11 and 5.12 are best possible.

One reason why the proofs of many of the bin-packing results are surprisingly complicated is because of the existence of examples like the following (due to Sylvia Halász [Ha]). This example gives a list L and a *sublist* $L' \subseteq L$ with $\mathrm{FFD}(L') > \mathrm{FFD}(L)$. Such behavior can be very annoying when one is trying to construct inductive proofs. ▣

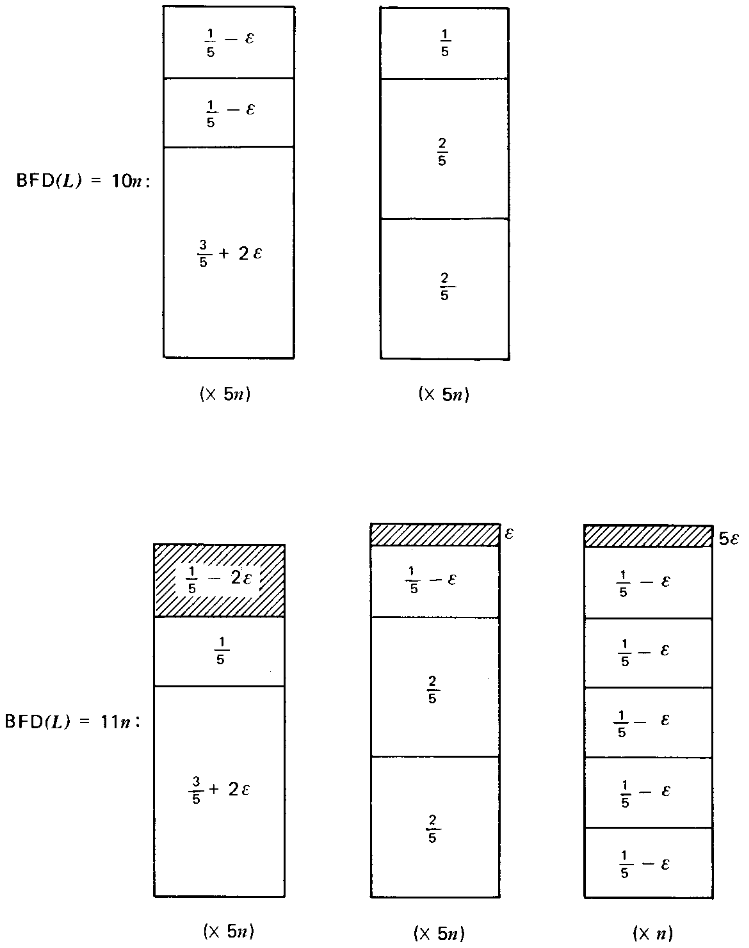


Figure 5.44 An example with L^* large and $\mathrm{BFD}(L)/\mathrm{FFD}(L) = 10/9$.

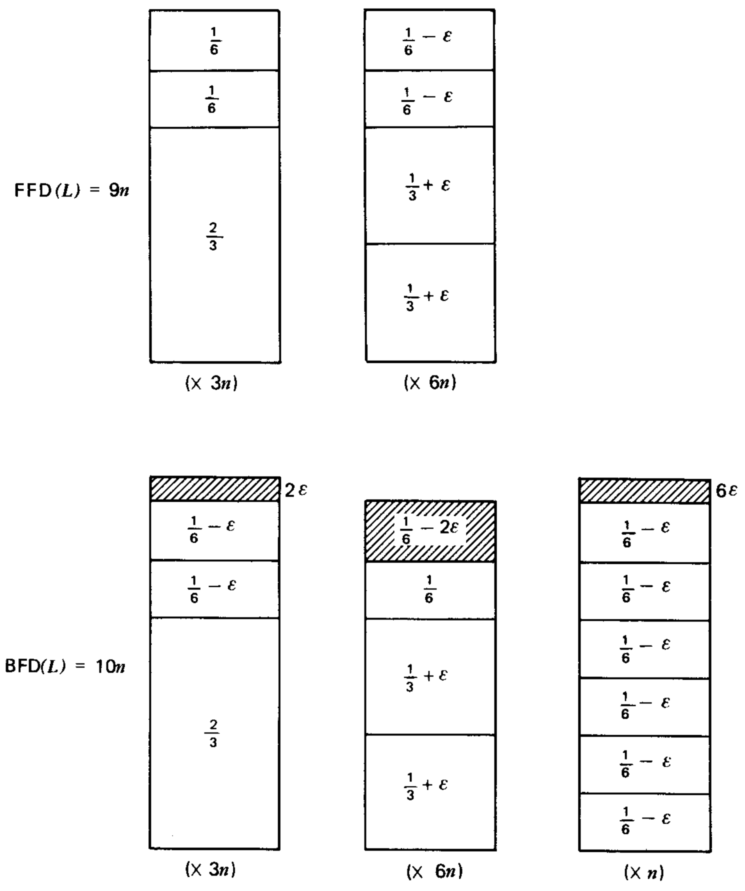


Figure 5.45 An example with L^* large and FFD(L)/BFD(L) = 11/10.

Example 17

$$L = (285, 188(\times 6), 126(\times 18), 115(\times 3), 112(\times 3), 75, \\ 60, 51, 12(\times 3), 10(\times 6), 9(\times 12))$$
$$L' = L - \{75\}$$

where $a(\times b)$ means b copies of a. The bins have capacity 396. The FFD-packings of L and L' are shown in Fig. 5.46.

For a rather complete discussion of numerous other bin-packing algorithms as well as comparisons of their *average* (as opposed to worst-case) behavior, the reader is referred to the doctoral dissertation of D. S. Johnson [J1], [J2].

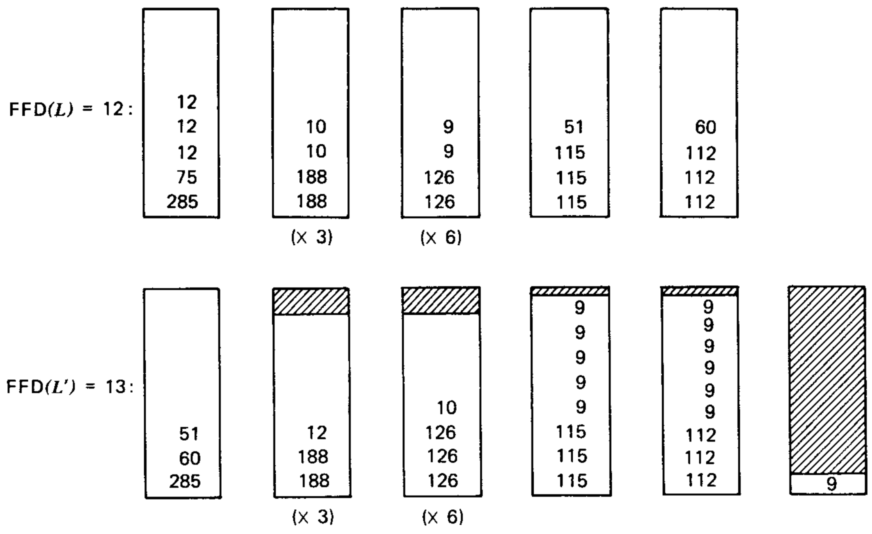


Figure 5.46 Example of $L' \subseteq L$ with $\mathrm{FFD}(L') < \mathrm{FFD}(L)$.

5.6 BOUNDS FOR SOME OTHER CASES

For the remainder of the chapter, we describe several recent results dealing with bounds on ω/ω_0 for other values of the parameters s, L, $<$, τ, and m.

Without the presence of a processor constraint, the scheduling problem with the parameters $s = 1$, all $\tau_i = 1$, $<$ empty, m, L arbitrary, is just ordinary bin packing for which we have obtained the asymptotic bound on ω/ω_0 of 17/10. When we have just a fixed number m of processors (possibly much smaller than the number n of tasks), this is equivalent to requiring that each bin contain at most m elements. For this situation, the following result of K. L. Krause applies.

Theorem [Kr] For $s = 1$, $\tau_i = 1$, $<$ empty, m, L arbitrary, we have

1. $\dfrac{\omega - 2}{\omega_0} < \dfrac{27}{10} - \dfrac{24}{10m}$;
2. There exist examples for which

$$\frac{\omega}{\omega_0} \geq \frac{27}{10} - \frac{37}{10m}$$

Hence the presence of the processor constraint contributes about 1 to the worst-case ratio bound.

If the tasks in L are arranged in the order of decreasing resource requirements, the resulting schedule is just that obtained by applying the FFD algorithm in the corresponding bin-packing problem, again with the additional restriction that no bin contains more than m weights. The finishing time is denoted here by ω_{FFD}.

Theorem [Kr] For $s = 1$, $<$ empty, all $\tau_i = 1$, m arbitrary, we have

$$\frac{\omega_{\mathrm{FFD}} - 1}{\omega_0} \leq 2 - \frac{2}{m} \qquad \text{for} \qquad m \geq 2$$

We next turn to several interesting results concerning the many-resource case (i.e., s is allowed to be arbitrary). Perhaps the most striking is the following.

Theorem 5.13 [GGJY] For $<$ empty, all $\tau_i = 1$, $m \geq n$, s, L arbitrary, we have

$$\omega \leq \left(s + \frac{7}{10}\right)\omega_0 + \frac{5}{2} \tag{31}$$

Furthermore, the coefficient of ω_0 is best possible.

Of course for $s = 1$ this is essentially just Theorem 5.9 (with a slightly weaker constant term). In fact, the inductive proof of Theorem 5.13 relies on Theorem 5.9 to begin the induction.

Comparing this result to that of Theorem 5.6, we see that the restriction that all $\tau_i = 1$ allowed the bound on ω/ω_0 to be strengthened from $s + 1$ to $s + \frac{7}{10}$. It had been previously shown by A. Yao [Y1] that asymptotically ω/ω_0 could not exceed $s + \frac{17}{20}$.

As might be suspected from Theorem 5.5, the restriction that all $\tau_i = 1$ for general $<$ is not sufficient to prevent the ratio ω/ω_0 from blowing up rather badly. The strongest results currently known here are the following.

Theorem 5.14 [GGJY] If all $\tau_i = 1$, $m \geq n$, s, $<$, L arbitrary,

1. $\dfrac{\omega}{\omega_0} \leq \dfrac{1}{2} s\omega_0 + \dfrac{1}{2} s + 1$
2. There exist examples for which

$$\frac{\omega}{\omega_0} \geq \frac{1}{2} s\omega_0 + \frac{1}{2} s + 1 - \frac{2s}{\omega_0}$$

If the schedule is formed by a critical path algorithm then the worst-case bound on ω/ω_0 improves considerably. In particular, the following result can be proved [GGJY].

Theorem 5.15 If all $\tau_i = 1$, $m \geq n$, s, $<$ arbitrary, we have

1. $\dfrac{\omega_{\mathrm{CP}}}{\omega_0} \leq \dfrac{17}{10} s + 1$;
2. For any $\varepsilon > 0$, there exist examples for which

$$\frac{\omega_{\mathrm{CP}}}{\omega_0} > \frac{17}{10} s + 1 - \varepsilon$$

Recall that in the special case $s = 0$, even with a processor constraint (i.e., allowing $m < n$), T. C. Hu has shown that $\omega_{\mathrm{CP}} = \omega_0$ (see Chapter 2).

There are several natural generalizations of the resource model we have been considering for which research work is just beginning. These include allowing processors with different speeds (see Chapter 2), prohibiting certain processors from executing certain tasks, and bin packing with bins of different capacities. Space considerations do not permit us to go into these topics here, however.