




UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA-CAMPAIGN
BOOKSTACKS



Digitized by the Internet Archive
in 2012 with funding from
University of Illinois Urbana-Champaign

330

STX

B385

1993:160 COPY 2

On Scheduling Parallel Machines with a Partially Shared Resource

THE LIBRARY OF THE
NOV 11 1993
OF URBANA-CHAMPAIGN

Narayan Raman
Department of Business Administration
University of Illinois

BEBR

FACULTY WORKING PAPER NO. 93-0160

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

September 1993

**On Scheduling Parallel Machines
with a Partially Shared Resource**

Narayan Raman
Department of Business Administration

On Scheduling Parallel Machines
with a Partially Shared Resource

Narayan Raman

Department of Business Administration
University of Illinois at Urbana-Champaign
Champaign, Illinois 61820

September 1993

ABSTRACT

This paper deals with the computational complexity of several scheduling problems arising in a class of parallel machine systems with a partially shared resource. The manufacturing shop comprises one operator and M identical machines operating in parallel. Each job in set \mathcal{J} needs to be scheduled on one of the machines. Before a job can be processed on any machine, it needs to be setup on that machine. During its setup, a job requires both the machine and the operator; however, only the machine is required for processing the job. We show that, for this system, optimizing many of the well-known scheduling measures results in NP-complete problems even when $M = 2$.

1 Introduction

In this paper, we consider scheduling problems arising in a system comprising one operator and M identical machines operating in parallel. Each of the N jobs in set \mathcal{J} needs to be scheduled on one of the machines. Before a job can be processed on any machine, it has to be setup on that machine. During its setup, a job requires both the machine and the operator; however, only the machine is required for processing the job. All jobs are available at time zero. No preemption is allowed during setup or processing, or between setup and processing.

This class of resource-constrained scheduling problems is gaining increasing importance because of its relevance in flexible manufacturing systems (FMSs), as well as in cellular manufacture. In an FMS, a robot serves the role of the operator that is shared among several pieces of equipment for the purpose of part setups and tool changes. Similarly, “one worker multiple machine systems” (Krajewski and Ritzman 1993) form an integral part of many group technology cells. The well-known case on *Fabritek Corporation* (Sasser et al. 1982) highlights the importance of minimizing operator-machine interference in these systems.

This paper deals with the computational complexity of the problems arising in this class of scheduling systems that have a partially shared resource. Note that the maximum completion time minimization problem for identical parallel machines is known to be NP-complete even in the absence of resource constraints. We show that this problem is rendered strongly NP-complete in the presence of a partially shared resource. Furthermore, the total completion time minimization problem that is solvable in polynomial time without resource constraints becomes NP-complete for this class of systems. We show that optimizing many of the other well-known scheduling measures results in NP-complete problems as well, and that all complexity results hold even for two-machine systems. [The reader is referred to Blazewicz et al. 1986 for a comprehensive discussion of the results currently available on the complexity of parallel machine scheduling problems.]

2 Problem Formulation

Let s_j , p_j , d_j , and w_j , respectively, denote the setup time, processing time, due date and weight of job j , $j \in \mathcal{J}$. Let C_j denote the completion time of job j in a schedule for any $j \in \mathcal{J}$. Then, the general system of constraints satisfied by any feasible schedule is given by

$$\sum_{m=1}^M x_{jm} = 1, \quad \forall j \quad (1)$$

$$y_{ijm} - x_{im}x_{jm} = 0, \quad \forall i, j, m \quad (2)$$

$$C_i \geq s_i + p_i, \quad \forall i \quad (3)$$

$$C_j - C_i - s_j - p_j + p_i \left(1 - \sum_{m=1}^M y_{ijm} \right) + L(1 - \delta_{ij}) \geq 0, \quad \forall i, j \quad (4)$$

$$C_i - C_j - s_i - p_i + p_j \left(1 - \sum_{m=1}^M y_{ijm} \right) + L\delta_{ij} \geq 0, \quad \forall i, j \quad (5)$$

$$C_j \geq 0, \quad \forall j; \quad x_{jm}, y_{ijm}, \delta_{ij} \in \{0, 1\}, \quad \forall i, j, m \quad (6)$$

where

$$x_{im} = \begin{cases} 1, & \text{if product } i \text{ is produced on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{ijm} = \begin{cases} 1, & \text{if jobs } i \text{ and } j \text{ are both produced on machine } m \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if } i \text{ precedes } j \text{ on the operator} \\ 0, & \text{otherwise} \end{cases}$$

and L is a large number. (1) insures that each job is assigned to exactly one machine. (2) defines y_{ijm} , while (3) requires that any job can finish only when its setup and processing is completed. (4) and (5) together insure that each machine, as well as the operator, works on at most one job at a time. Finally, (6) specifies the nature of the variables.

For this system of schedules, we consider the following problems: i) **CMAX**: minimizing maximum completion time $C_{max} = \max_j \{C_j\}$; ii) **TCT**: minimizing total completion time $C_{tot} = \sum_j C_j$, and **WCT**: minimizing total weighted completion time $C_w = \sum_j w_j C_j$; iii) **LMAX**: minimizing maximum lateness $L_{max} = \max_j \{C_j - d_j\}$; iv) **TTD**: minimizing total tardiness $T_{tot} = \sum_j \max(0, C_j - d_j)$, and **WTD**: minimizing total weighted tardiness $T_w = \sum_j w_j \max(0, C_j - d_j)$; and v) **TER**: minimizing total earliness $E_{tot} = \sum_j \max(0, d_j - C_j)$ subject to $C_i \leq d_i, \forall i$, and **WER**: minimizing total weighted earliness $E_w = \sum_j w_j \max(0, d_j - C_j)$ subject to $C_i \leq d_i, \forall i$.

3 NP-Completeness Proofs

Because all complexity results hold even for two-machine systems, we restrict ourselves throughout this section to such systems only.

Theorem 1. *CMAX is NP-complete in the strong sense.*

PROOF: **CMAX** clearly lies in *NP*. We show that the 3-PARTITION problem which is known to be NP-complete in the strong sense (Garey and Johnson 1979) is reducible to **CMAX**. Consider an arbitrary instance of 3-PARTITION given by a set \mathcal{Q} of $3q$ elements of size a_i for each $i \in \mathcal{Q}$, and a positive integer B such that i) $B/4 < a_i < B/2$, $\forall i \in \mathcal{Q}$, and ii) $\sum_{i \in \mathcal{Q}} a_i = qB$. The recognition problem is stated as: Is there a partition of \mathcal{Q} into q disjoint subsets $(\mathcal{Q}_1, \mathcal{Q}_2, \dots, \mathcal{Q}_q)$ such that $\sum_{i \in \mathcal{Q}_j} a_i = B$, for $1 \leq j \leq q$?

We construct the equivalent instance of **CMAX** with three types of jobs. Let \mathcal{J}_i denote the set of jobs of type i , $i = 1, 2, 3$, and let $\mathcal{J}_1 = \{1, \dots, 3q\}$, $|\mathcal{J}_2| = |\mathcal{J}_3| = q$. Define $P = B(q + 1) + 1$. Assign job parameters as follows:

$$s_i = 0, \quad p_i = a_i, \quad i = 1, \dots, 3q,$$

$$s_i = P - B, \quad p_i = 0, \quad i \in \mathcal{J}_2, \text{ and}$$

$$s_i = B, \quad p_i = P - B, \quad i \in \mathcal{J}_3.$$

Given an instance of 3-PARTITION, this instance of **CMAX** can be constructed in polynomial time. We show that for this instance, the maximum completion time (makespan) $C_{max} \leq qP$, if and only if 3-PARTITION has a solution.

First suppose that 3-PARTITION has a solution. The required solution for **CMAX** is constructed as follows. Sequence type 3 and type 2 jobs on the operator such that type 3 jobs are in positions $1, 3, \dots, 2q - 1$, and type 2 jobs are in positions $2, 4, \dots, 2q$. Assign all type 3 jobs to machine M2, and type 2 jobs to machine M1 as shown in Figure 1.

INSERT FIGURE 1 HERE

Note that all type 2 and type 3 jobs are completed by time qP . Furthermore, M1 has q slots of idle time, each of duration B . Clearly, if 3-PARTITION has a solution, then \mathcal{J}_1 can be partitioned into

q subsets, each with a total processing time of B . These subsets are inserted into the idle times slots on machine M1 to yield the desired solution.

Next, we show that if $C_{max} \leq qP$, then 3-PARTITION has a solution. First, note that qP is a lower bound on C_{max} because the total work to be done on both machines is $\sum_{i \in \mathcal{J}_1} a_i + 2q(P - B) + qB = 2qP$. Hence, $C_{max} = qP$, and there is no idle time on either machine through the makespan. Also, note that the total setup time is $q(P - B) + qB = qP$; therefore, the operator has no idle time either.

Lemma 1. *If $C_{max} \leq qP$, then the operator must process type 3 jobs in positions $1, 3, \dots, 2q - 1$, and type 2 jobs in positions $2, 4, \dots, 2q$.*

PROOF: Since the operator is busy throughout the duration of the makespan qP , the job set up last by the operator must be of type 2 because it has zero processing time. If the job in position $2q - 1$ is not of type 3, then two type 2 jobs are processed together. This implies that the setup of the last type 3 job must be completed by time $qP - 2(P - B)$, and its processing should be completed by time

$$\tau = qP - 2(P - B) + (P - B) = qP - (P - B).$$

By τ , the two machines should also finish $(q - 1)$ type 2 jobs. Hence, the total work done by both machines till time τ is

$$W = qP + (q - 1)(P - B) = 2Bq^2 + 2q - 1.$$

But the total capacity available on these two machines until τ is

$$CAP = 2\tau = 2P(q - 1) + 2B = 2Bq^2 + 2(q - 1) < W$$

which is infeasible. Hence, the job in position $2q - 1$ must be of type 3, and it must be taken up for setup at time $qP - P$. This implies that the remaining $(q - 1)$ jobs of type 2 and type 3 each must have completed setup by time $qP - P = P(q - 1)$. Setting $q \leftarrow q - 1$, and repeating the above arguments recursively completes the proof of the lemma. \square

It follows from the above proof that the type 2 job in position $2i - 1$ and the type 3 job in position $2i$, $i = 1, \dots, q$, must be run on different machines. As shown in Figure 1, this leaves q idle time slots of length B each, and for all type 1 jobs to be completed by qP , it must be possible to complete

processing them during these slots. But this implies that \mathcal{J}_1 can be partitioned into q subsets of size B each, which in turn implies that 3-PARTITION has a solution. \square

Corollary 1. *LMAX is NP-complete in the strong sense.*

PROOF: In the instance of **CMAX** generated above, assign $d_i = 0$, $i \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3$. Then, for this instance, $L_{max} \leq qP$, if and only if $C_{max} \leq qP$. \square

Theorem 2. *TCT is NP-complete.*

PROOF: **TCT** clearly lies in *NP*. We show that the SUBSET SUM problem that is known to be NP-complete (Garey and Johnson 1979) is reducible to **TCT**. Consider an arbitrary instance of SUBSET SUM defined by a positive integer B' and a set \mathcal{Q}' of q elements of (integer) size a'_i , $i \in \mathcal{Q}'$. The recognition problem is stated as: Is there a subset $\mathcal{Q}'_0 \subseteq \mathcal{Q}'$ such that $\sum_{i \in \mathcal{Q}'_0} a'_i = B'$?

Let $a, b \geq 1$ be arbitrary integers such that a is not an integer multiple of b . Given the original instance of SUBSET SUM, we generate an equivalent instance by defining a set \mathcal{Q} of q elements of size $a_i = ba'_i$, $i = 1, \dots, q$, as well as an integer $B = bB'$. This transformation can be done in polynomial time; clearly, there exists a subset $\mathcal{Q}'_0 \subseteq \mathcal{Q}'$ for the original instance with the property that $\sum_{i \in \mathcal{Q}'_0} a'_i = B'$ if and only if there is a subset $\mathcal{Q}_0 \subseteq \mathcal{Q}$ such that $\sum_{i \in \mathcal{Q}_0} a_i = B$ for the modified problem. Note that this transformation insures that

Remark 1. *There exists no subset $\mathcal{Q}_1 \subseteq \mathcal{Q}$ with the property that $\sum_{i \in \mathcal{Q}_1} a_i = kb - a$ for any integer k .*

In the rest of the proof, we restrict ourselves to the modified version of SUBSET SUM. Let $a_{max} = \max_{i \in \mathcal{Q}} \{a_i\}$, and $\sum_{i \in \mathcal{Q}} a_i = A$. We construct the corresponding instance of **TCT** by considering jobs I , J and K , as well as job sets $\mathcal{E} = \{1, \dots, q\}$, $\mathcal{F} = \{q + 1, \dots, q + f\}$, \mathcal{G} and \mathcal{H} such that $|\mathcal{G}| = g$, and $|\mathcal{H}| = h$ with the following parameters.

<i>Job</i>	<i>Processing Time</i>	<i>Setup Time</i>
<i>I</i>	$p_I = 0$	$s_I = a$
<i>J</i>	$p_J = a$	$s_J = 1$
<i>K</i>	$p_K = 0$	$s_K = \sigma_1$
$j \in \mathcal{E}$	$p_j = \pi_1 + a_j$	$s_j = 0$
$j \in \mathcal{F}$	$p_j = \pi_1$	$s_j = 0$
$j \in \mathcal{G}$	$p_j = \pi_2$	$s_j = 0$
$j \in \mathcal{H}$	$p_j = 0$	$s_j = \sigma_2$

where

$$f = q(q + 1)a_{max}$$

$$\pi_1 = (f + 1)(B + a) + f$$

$$\tau = (q + f)\pi_1 + A + a$$

$$D_0 = \frac{1}{2}(q + f)(q + f + 1)\pi_1 + (q + 1)\pi_1$$

$$\theta = q\pi_1 + B + a$$

$$\sigma_1 = q\pi_1 + B - 1$$

$$D_1 = \sigma_1 + \theta$$

$$g = D_0 + D_1$$

$$\pi_2 = g(\tau + 1)$$

$$D_2 = g\tau + \frac{1}{2}g(g + 1)\pi_2$$

$$h = D_0 + D_1 + D_2$$

$$\sigma_2 = h(\theta + 1)$$

$$D = \sigma_2 + \frac{1}{2}h(h + 1)\sigma_2.$$

Given an instance of SUBSET SUM, this instance of **TCT** can be constructed in polynomial time.

We show that for this instance, the total completion time $C_{tot} \leq D$, if and only if SUBSET SUM has a solution. Note that Remark 1 implies that

Remark 2. *There exists no subset $\mathcal{S} \subseteq \mathcal{E} \cup \mathcal{F}$, with the property that $\sum_{i \in \mathcal{S}} a_i = kb - a$ for any integer k .*

If SUBSET SUM has a solution, then there must exist a subset $\mathcal{E}_0 \subseteq \mathcal{E}$ corresponding to Q_0 such that $\sum_{j \in \mathcal{E}_0} p_j = e_0\pi_1 + B$, where $e_0 = |\mathcal{E}_0|$. Let $\mathcal{F}_0 = \{q + j | j \in \mathcal{E} \setminus \mathcal{E}_0\} \subseteq \mathcal{F}$. The desired schedule for **TCT** is constructed as shown in Figure 2.

INSERT FIGURE 2 HERE

Note that jobs in set \mathcal{G} start being processed on machine M1 at time

$$\sum_{j \in \mathcal{E} \cup \mathcal{F}} p_j + s_I = (q + f)\pi_1 + A + a = \tau,$$

and jobs in set \mathcal{H} start being set up on machine M2 at time

$$s_K + s_J + p_J = q\pi_1 + B - 1 + (1 + a) = \theta.$$

Also note that

$$\begin{aligned} \sum_{j \in \mathcal{F}_0 \cup \mathcal{E}_0} C_j &\leq \sum_{j=1}^q j(\pi_1 + a_{max}) \\ C_I &= q\pi_1 + B + a = \theta \\ \sum_{j \in \{\mathcal{F} \setminus \mathcal{F}_0\} \cup \{\mathcal{E} \setminus \mathcal{E}_0\}} C_j &\leq \sum_{j=q+1}^{q+f} (j\pi_1 + B + a) + \sum_{j=1}^q ja_{max} \\ \sum_{j \in \mathcal{G}} C_j &= \sum_{j=1}^g (\tau + j\pi_2) = g\tau + \frac{1}{2}g(g+1)\pi_2. \end{aligned}$$

Hence, the sum of completion times of all jobs on machine M1 is

$$\begin{aligned} C(1) &= \sum_{j \in \mathcal{F}_0 \cup \mathcal{E}_0} C_j + C_I + \sum_{j \in \{\mathcal{F} \setminus \mathcal{F}_0\} \cup \{\mathcal{E} \setminus \mathcal{E}_0\}} C_j + \sum_{j \in \mathcal{G}} C_j \\ &\leq \frac{1}{2}(q+f)(q+f+1)\pi_1 + q(q+1)a_{max} + q\pi_1 + B + a + f(B+a) + g\tau + \frac{1}{2}g(g+1)\pi_2 \\ &= \frac{1}{2}(q+f)(q+f+1)\pi_1 + (q+1)\pi_1 + g\tau + \frac{1}{2}g(g+1)\pi_2 \\ &= D_0 + D_2. \end{aligned} \tag{7}$$

And the sum of completion times of all jobs on machine M2 is

$$\begin{aligned} C(2) &= C_K + C_J + \sum_{j \in \mathcal{H}} C_j \\ &= \sigma_1 + \theta + \sum_{j=1}^h (\theta + j\sigma_2) \\ &= \sigma_1 + \theta + h\theta + \frac{1}{2}h(h+1)\sigma_2 \\ &= D_1 + h\theta + \frac{1}{2}h(h+1)\sigma_2. \end{aligned} \tag{8}$$

Hence,

$$\begin{aligned}
C_{tot} = C(1) + C(2) &\leq D_0 + D_1 + D_2 + h\theta + \frac{1}{2}h(h+1)\sigma_2 \\
&= h(\theta+1) + \frac{1}{2}h(h+1)\sigma_2 \\
&= D
\end{aligned}$$

as required. Next we show that if $C_{tot} \leq D$, then SUBSET SUM must have a solution.

Lemma 2. $C_i > C_j$, for any $i \in \mathcal{H}$, $j \notin \mathcal{H}$. Furthermore, the processing of jobs in \mathcal{H} must start at θ .

PROOF: If there is a job $j \notin \mathcal{H}$ that is completed after some job in \mathcal{H} , then $C_{tot} > C_j + \sum_{i \in \mathcal{H}} C_i \geq \sigma_2 + \frac{1}{2}h(h+1)\sigma_2 = D$ which contradicts the original assumption. If the processing of \mathcal{H} starts later than θ , then

$$C_{tot} > \sum_{j \in \mathcal{H}} C_j \geq h(\theta+1) + \frac{1}{2}h(h+1) = D$$

which is again a contradiction. But the total setup time required on all jobs not in \mathcal{H} equals

$$s_I + s_J + s_K = a + 1 + (q\pi_1 + B - 1) = \theta.$$

Because these jobs precede all jobs in \mathcal{H} , the latter cannot start before θ either. Hence, the processing of jobs in \mathcal{H} must start at θ . \square

Two results follow from the above proof.

Corollary 2. *There is no idle time on the operator through the makespan of the problem.*

Corollary 3. *Jobs in \mathcal{H} are processed without any intervening idle time.*

Consequently, we have

$$\sum_{j \notin \mathcal{H}} C_j \leq D - \sum_{j \in \mathcal{H}} C_j = D - \left\{ h\theta + \frac{1}{2}h(h+1)\sigma_2 \right\} = h \quad (9)$$

From Lemma 2, we assume without loss of generality that all jobs in \mathcal{H} are processed on the same machine; let this machine be M2.

Lemma 3. $C_i > C_j$, for any $i \in \mathcal{G}$, $j \notin \mathcal{G} \cup \mathcal{H}$. Furthermore, the processing of jobs in \mathcal{G} must start at τ .

PROOF: The proof is similar to that of Lemma 2 above. If there is a job $j \notin \mathcal{G} \cup \mathcal{H}$ that finishes after some job in \mathcal{G} , then $\sum_{i \notin \mathcal{H}} C_i > C_j + \sum_{i \in \mathcal{G}} C_i \geq \pi_2 + \frac{1}{2}g(g+1)\pi_2 = g\tau + g + \frac{1}{2}g(g+1)\pi_2 = D_0 + D_1 + D_2 = h$ which contradicts (9). If the processing of \mathcal{G} starts later than τ , then

$$\sum_{j \notin \mathcal{H}} C_j > \sum_{j \in \mathcal{G}} C_j \geq g(\tau + 1) + \frac{1}{2}g(g+1)\pi_2 = h$$

which again contradicts (9). But the total processing and setup times required on all jobs not in $\mathcal{G} \cup \mathcal{H}$ is

$$\sum_{j \in \mathcal{E} \cup \mathcal{F}} p_j + s_K + s_I + s_J + p_J = (q+f)\pi_1 + A + (q\pi_1 + B - 1) + a + 1 + a = \tau + \theta. \quad (10)$$

From Lemma 2, it follows that jobs in \mathcal{G} cannot start processing before τ either. Hence, the processing of jobs in \mathcal{G} must start at τ . \square

We make three observations here. First from Lemmas 2 and 3, and (10), it follows that

Remark 3. *Jobs I , J , and K as well as jobs in \mathcal{E} and \mathcal{F} must be completed without any intervening idle time.*

Second, because $\tau > \theta$, \mathcal{G} and \mathcal{H} must be processed on different machines, i.e., \mathcal{G} must be processed on machine M1. Finally, from the proof of Lemma 3, it follows that

Corollary 4. *Jobs in \mathcal{G} are processed without any intervening idle time.*

Consequently, from (9), we have

$$\sum_{j \notin \mathcal{G} \cup \mathcal{H}} C_j \leq h - \sum_{j \in \mathcal{G} \cup \mathcal{H}} C_j = h - \left\{ g\tau + \frac{1}{2}g(g+1)\pi_2 \right\} = g. \quad (11)$$

Next we establish the relative order of jobs I , J and K .

Lemma 4. *J cannot be the first job taken up for setup.*

PROOF: From Corollary 2, if J is taken up first by the operator on (say) M1, then the next setup must take place after one time unit on M2 because of the subsequent processing required on J . This induces an idle time of one time unit on M2 because no job from $\mathcal{E} \cup \mathcal{F} \cup \{I\} \cup \{K\}$ can be processed during this interval, and a contradiction of Remark 3 results. \square

Lemma 5. *I , J and K must complete processing by θ .*

PROOF: From Lemma 2, the setup of jobs I , J and K must be completed by θ . The only schedule in which their processing will not complete by θ is one in which J is taken up last among these three jobs for setup as depicted in Figure 3.

INSERT FIGURE 3 HERE

However, in this case, the gap $\tau - (\theta + a)$ must be filled by jobs from $\mathcal{E} \cup \mathcal{F}$ to prevent any idle time. Therefore, there must exist a subset $\mathcal{S} \subseteq \mathcal{E} \cup \mathcal{F}$ such that

$$\begin{aligned} \sum_{j \in \mathcal{S}} p_j &= \tau - (\theta + a) \\ &= (q + f)\pi_1 + A + a - (q\pi_1 + B + a) - a \\ &= (f\pi_1 + A - B) - a. \end{aligned}$$

But this contradicts Remark 2 since $(f\pi_1 + A - B)$ is an integer multiple of b . Hence, this schedule is not feasible and the proof is complete. \square

From Lemmas 4 and 5, it follows that the setup of I and the processing of J must occur in parallel, and the setup sequence followed by the operator must be $K - J - I$ in the interval $[0, \theta]$. This implies that the setup of I must start at $\theta - a$, and $C_I = C_J = \theta$. From (11), it follows that

$$\begin{aligned} C_I + \sum_{j \in \mathcal{E} \cup \mathcal{F}} C_j &\leq g - C_K - C_J \\ &= D_0 + D_1 - \sigma_1 - \theta \\ &= D_0 \end{aligned} \tag{12}$$

Lemma 6. *Exactly q jobs from $\mathcal{E} \cup \mathcal{F}$ must be completed by time $\theta - a$.*

PROOF: Let n be the number of jobs from $\mathcal{E} \cup \mathcal{F}$ that are completed by $\theta - a$. If $0 \leq n \leq q - 1$, then

$$\begin{aligned} C_I + \sum_{j \in \mathcal{E} \cup \mathcal{F}} C_j &\geq \theta + \sum_{j=1}^n j\pi_1 + \sum_{j=n+1}^{q+f} \{(j - n)\pi_1 + \theta\} \\ &= \frac{1}{2}(q + f)(q + f + 1)\pi_1 + \theta + (q + f - n)(\theta - n\pi_1) \\ &= D_0 - (q + 1)\pi_1 + (q + f - n)(\theta - n\pi_1) + \theta \\ &> D_0 \end{aligned}$$

which contradicts (12). If $q + 1 \leq n \leq q + f$, then

$$\begin{aligned}
C_I + \sum_{j \in \mathcal{E} \cup \mathcal{F}} C_j &\geq \sum_{j=1}^n j\pi_1 + \{n\pi_1 + a\} + \sum_{j=n+1}^{q+f} (j\pi_1 + a) \\
&= \frac{1}{2}(q+f)(q+f+1)\pi_1 + n\pi_1 + a + (q+f-n)a \\
&= D_0 + \pi_1 \{n - q - 1\} + a + (q+f-n)a \\
&> D_0
\end{aligned}$$

which again contradicts (12) and the proof is complete. \square

From the processing times of jobs in sets \mathcal{E} and \mathcal{F} , it follows that there must be a subset $\mathcal{S}_0 \subseteq \mathcal{E} \cup \mathcal{F}$ such that $|\mathcal{S}_0| = q$, and

$$q\pi_1 + \sum_{j \in \mathcal{S}_0} a_j = \theta - a$$

or

$$\sum_{j \in \mathcal{S}_0} a_j = B.$$

But this implies that SUBSET SUM has a solution and the proof is complete. \square

Theorem 3. *TTD is NP-complete.*

PROOF: We reduce the single machine total tardiness problem (SMT) that is known to be NP-complete (Du and Leung 1990) to **TTD**. Let the instance of SMT be given by a set of n jobs with processing times and due dates of π_i and δ_i , $i = 1, \dots, n$, respectively. Let c_i denote the completion time of job i , $i = 1, \dots, n$ in any schedule. The recognition problem is stated as: Is there a permutation of these n jobs such that $\sum_{j=1}^n \max(0, c_j - \delta_j) \leq Z$?

Let $\gamma = n(Z + 1) + \sum_{i=1}^n \max(\delta_i, \pi_i)$. The equivalent instance of **TTD** comprises $n + Z + 1$ jobs with the following parameters:

$$s_i = 0, \quad p_i = \pi_i, \quad d_i = \delta_i, \quad i = 1, \dots, n, \text{ and}$$

$$s_{n+i} = \gamma, \quad p_{n+i} = 0, \quad d_{n+i} = i\gamma, \quad i = 1, \dots, Z + 1.$$

We show that for this instance, $T_{tot} \leq Z$ if and only if SMT has a solution. If SMT has a solution, then the corresponding schedule for **TTD** is obtained by sequencing jobs $n + 1$ through $n + Z + 1$ on M2 in the order of their indexes. Jobs 1 through n are scheduled on M1 such that $C_{[i]} = c_{[i]}$, $i = 1, \dots, n$ to yield the required solution.

If **TTD** has a solution, then in a manner similar to the proof of Lemma 2, it can be established that i) $C_i > C_j$, for any $i \in \{n+1, \dots, n+Z+1\}$, and $j \in \{1, \dots, n\}$, and ii) jobs in $\{n+1, \dots, n+Z+1\}$ must be setup in the order of their indexes, starting at time zero and without any intervening idle time. Without any loss of generality, we may assume that jobs in $\{n+1, \dots, n+Z+1\}$ are setup on machine M2. It is easy to see that all these jobs are completed exactly on time, and therefore, incur no tardiness. But this implies that $\sum_{j=1}^n \max(0, C_i - d_i) \leq Z$ and SMT has a solution. \square

Theorem 4. **TER** is NP-complete.

PROOF: We show that that the single machine total earliness problem (SER) that is known to be NP-complete (Chand and Schneeberger 1986) can be reduced to **TER**. Let the instance of SER be given by a set of n jobs with processing times and due dates of π_i and δ_i , $i = 1, \dots, n$, respectively. Let c_i denote the completion time of job i , $i = 1, \dots, n$ in any schedule. The recognition problem is stated as: Is there a permutation of these n jobs such that $c_i \leq \delta_i, \forall i$, and $\sum_{j=1}^n \max(0, \delta_i - c_i) \leq Z$?

The equivalent instance of **TER** comprises $n + 1$ jobs with the following parameters:

$$s_i = 0, \quad p_i = \pi_i, \quad d_i = \delta_i, \quad i = 1, \dots, n, \text{ and}$$

$$s_{n+1} = d_{n+1} = \sum_{i=1}^n d_i + 1, \quad p_{n+1} = 0.$$

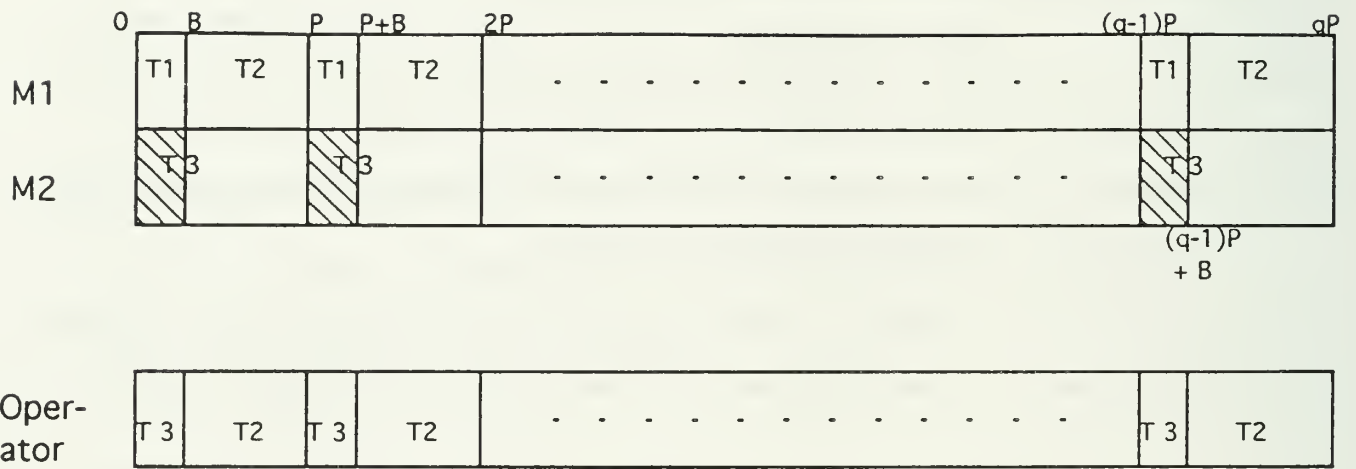
Given these parameters, note that $C_i \leq d_i, i = 1, \dots, n+1$, is insured only if job $(n+1)$ is scheduled on (say) machine M2 starting at time zero, and the other n jobs are scheduled on machine M1. It is easy to see that $E_{tot} \leq Z$ for this instance of **TER** if and only if SMT has a solution. \square

Because **TCT**, **TTD**, and **TER** are special cases of **WCT**, **WTD**, and **WER**, respectively, it follows that

Corollary 5. **WCT**, **WTD**, and **WER** are NP-complete.

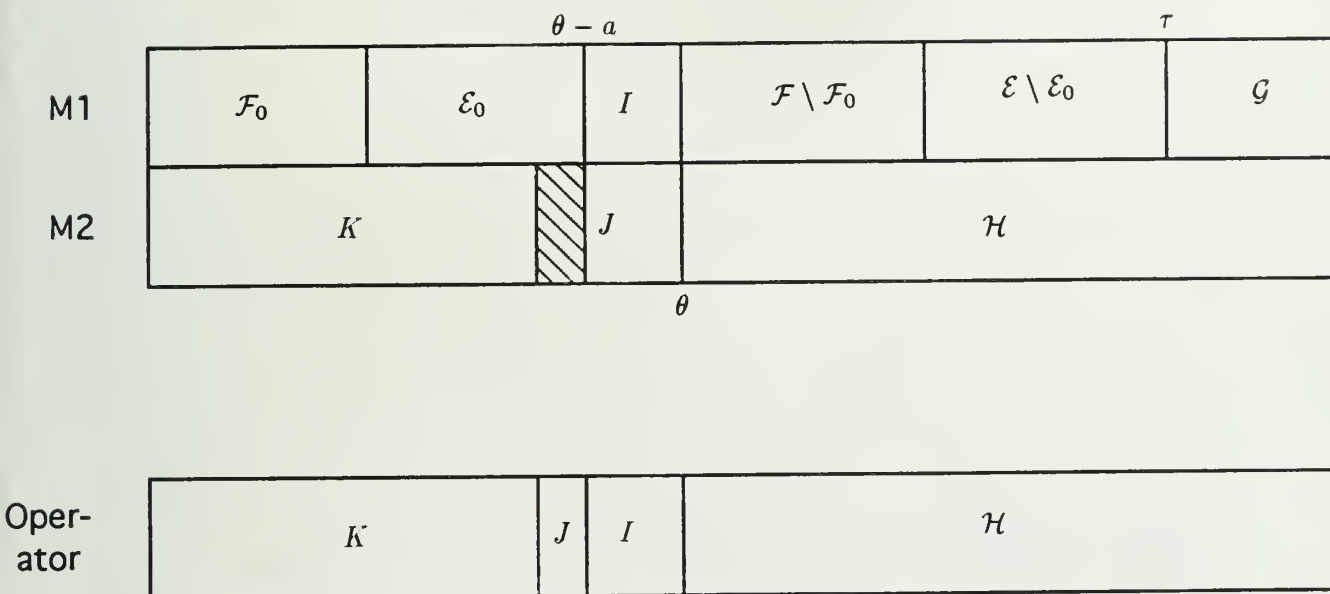
References

1. Blazewicz, J., W. Cellary, R. Slowanski and J. Weglarz (1986), *Scheduling Under Resource Constraints – Deterministic Models*, J. C. Baltzer AG, Basel, Switzerland.
2. Chand, S. and H. Schneeberger (1986), “A Note on the Single Machine Scheduling Problem with Minimum Weighted Completion Time and Maximum Allowable Tardiness,” *Naval Research Logistics Quarterly*, Vol. 33, 551–557.
3. Du, Z. and J. Y.-T. Leung (1990), “Minimizing Total Tardiness on One-Machine is NP-Hard,” *Mathematics of Operations Research*, Vol. 15, 483–495.
4. Garey, M. R. and D. S. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, NY.
5. Krajewski, L. J. and L. P. Ritzman (1993), *Operations Management – Strategy and Analysis*, Addison-Wesley Publishing Company, New York, NY.
6. Sasser, W. E., K. B. Clark, D. A. Garvin, M. B. W. Graham, R. Jaikumar and D. H. Maister (1982), *Cases in Operations Management: Analysis and Action*, Irwin, Homewood, IL.



T_i represents a job of type i , $i = 1, 2, 3$. The shaded area denotes the setup of a type 3 job.

Figure 1 - Proof of Theorem 1



The shaded area denotes the setup of job J .

Figure 2 - Proof of Theorem 2

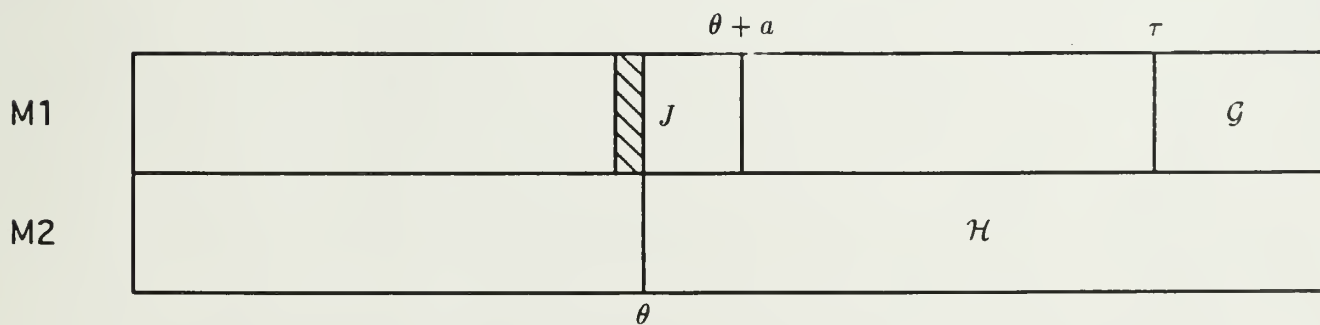


Figure 3 - Proof of Lemma 5

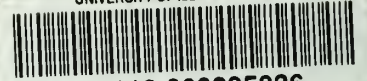
HECKMAN
BINDERY INC.



JUN 95

Bound-To-Pleas[®] N. MANCHESTER,
INDIANA 46962

UNIVERSITY OF ILLINOIS-URBANA



3 0112 060295836