Journal of the Operations Research
Society of Japan
Vol. 25, No. 1, March 1982

ON THE USE OF DECOMPOSITION APPROACHES IN A SINGLE MACHINE SCHDULING PROBLEM

Jeyaveerasingam George Shanthikumar
Syracuse University

John Alan Buzacott
University of Tronto

(Received January 6, 1981; Revised June 26, 1981)

Abstract A condition is given to decompose a large scheduling problem into two or more scheduling problems of smaller size. The applicability of the decomposition approach to scheduling problems is illustrated through the single machine scheduling problem $n/1/F_w|T_{max} = 0$, minimizing the weighted flow time with zero maximum tardiness, and a branch and bound algorithm incorporating the decomposition principle is presented to obtain the optimal schedule. This algorithm is then extended to obtain the optimal schedule to the single machine scheduling problem $n/1/F_w|T_{max}$, minimizing the weighted flow time with minimum/maximum tardiness, by reducing $n/1/F_w|T_{max}$ to $n/1/F_w|T_{max} = 0$.

1. Introduction

The idea of decomposing a problem to simplify its solution is not new. However in recent years the use of large-scale computers has led to rapid expansion in the use of decomposition techniques for optimization, for solution of reliability and electrical network problems, for process control and for a wide variety of other problems. In spite of the diversity of opinion on the definition of decomposition, the idea of decomposition can be in general stated as: a large complex system representing interacting elements could be broken up into subproblems of lower dimensionality. Presumably these subproblems can be treated independently for the purpose of optimization, control, design and so forth, and the coordination of the solution of these subproblems is the solution of the large original problem.

The most effective type of decomposition is to form disjoint subsystems, that is to form subsets of relations that do not contain any common variables so that each subset can be treated independently.

In this paper we will extend the idea of decomposition to permutation scheduling problems. In a permutation scheduling problem $P \triangleq \{J, \phi\}$, it is required to obtain the optimal permutation schedule of the n jobs in the given set $J \triangleq \{1,2,\ldots,n\}$ with an objective function ϕ. That is, we wish to minimize $\phi(s)$ over all possible permutation schedule s of the jobs $1,2,\ldots,n$.

In the next section we will present a sufficient condition to decompose a large permutation scheduling problem into two or more disjoint scheduling problems of smaller size. Following a brief introduction to the single machine scheduling problem in section 3, decomposability of the single machine scheduling problem $n/1/F_w|T_{\max} = 0$, minimizing the weighted flow time F_w with zero maximum tardiness, will be illustrated and a branch and bound algorithm to obtain the optimal schedule will be presented in section 4. In section 5, the branch and bound algorithm given in section 4 will be extended to the single machine scheduling problem $n/1/F_w|T_{\max}$ minimizing the weighted flow time with minimum maximum tardiness $T_{\max}$. For this we will show that $n/1/F_w|T_{\max}$ can be reduced to $n/1/F_w|T_{\max} = 0$.

2. Complete Decomposition

In this section we will present a condition and show that if the permutation scheduling P satisfies it, then P can be decomposed into two or more disjoint subsets of smaller size.

Condition 1:

If we can partition the set J into two or more (say m+1) mutually exclusive subsets $J(i)$, i=1,2,...,m+1, such that

$$\bigcup_{i=1}^{m+1} J(i) = J,$$

1.1 all jobs $j \in J(i)$ precede all jobs $k \in J(i-1)$ in an optimal schedule, $i=2,\ldots,m+1$,

and 1.2 the cost $\phi(s)$ of the schedule s of the n jobs can be expressed explicitly as a monotonically nondecreasing function $h(\cdot)$ of the individual costs $f_i(\pi_i)$ of the schedule π_i of the jobs in $J(i)$, $i=1,2,\ldots,m+1$, such that

$f_i(\pi_i)$ is independent of π_j, $j=1,2,\ldots,m+1, \neq i$, $i=1,2,\ldots,m+1$,

$$s = \bigcup_{i=0}^{m} \pi_{m+1-i} = \langle \pi_{m+1}, \pi_m, \ldots, \pi_1 \rangle$$

and $\phi(s) = h(f_i(\pi_i),\ i=1,2,\ldots,m+1)$,

then we have,

Theorem 2.1. If π_i^* is the optimal schedule for the subproblem $P(i) \triangleq \{J(i),\ f_i\}$ for every $i=1,2,\ldots,m+1$, such that $f_i(\pi_i^*)$ is the minimum of $f_i(\pi_i)$ over all possible permutation schedules of jobs in $J(i)$, then the optimal schedule s^* for problem P can be obtained by the recomposition rule

$$s^* = \bigcup_{i=0}^{m} \pi^*_{m+1-i} = <\pi^*_{m+1},\ \pi^*_m, \ldots, \pi^*_1> .$$

The optimal cost of the schedule s^* is

$$\phi(s^*) = h(f_i(\pi_i^*),\ i=1,2,\ldots,m+1) .$$

Proof: From condition 1.1 we know that at least one schedule formed by the recomposition of the schedules of the subsets $J(i)$, $i = m+1,\ldots,1$, should be optimal. Let this schedule be $s' = <\pi'_{m+1},\ \pi'_m, \ldots, \pi'_1>$. Then $\phi(s') \le \phi(s)$ for all possible schedules s of the jobs in J. Since $h(\cdot)$ is minimum when $\pi_i = \pi_i^*$, $i=1,2,\ldots,m+1$, from condition 1.2 we have $\phi(s') = h(f_i(\pi'_i),\ i=1,2,\ldots,m+1) \ge h(f_i(\pi_i^*),\ i=1,2,\ldots,m+1) = \phi(s^*)$. Then obviously $s^* = <\pi^*_{m+1},\ \pi^*_m,\ldots,\pi^*_1>$ is an optimal schedule and the minimum cost is $\phi(s^*)$.

Q.E.D.

If the above condition is satisfied, a large scheduling problem can be decomposed into two or more disjoint subsets of smaller problems. These subproblems can be solved separately and the optimal schedule for the large problem can be obtained by recomposing them as given above. In some large scheduling problems a feasible schedule may be found if and only if it satisfies condition 1.1. Then we say that this condition is the feasibility condition. In other cases this will be called the optimality condition.

The set of jobs can be sometimes partitioned into subsets based either on the feasibility or on the optimality condition. In this paper we will illustrate how this job partitioning may be carried out based on the feasibility condition. For this we will solve the one machine scheduling problem minimizing the weighted sum of flow time with the constraint of maintaining all jobs early (non tardy). Symbolically the problem is $n/1/F_w|T_{\max} = 0$.

3. One Machine Scheduling Problem

The problem of optimally scheduling a set $J = \{1,2,\ldots,n\}$ of n jobs on one machine with criteria C will be written $n/1/C$. Associated with each job

$i \in J$ are its processing time p_i, due-date d_i and its weight w_i. All jobs will be assumed to be available at time $t=0$ and not be preempted by any other job. A schedule or sequence S produces a completion flow time $F_i = F_i(S)$ for each job i and consequently we have a tardiness $T_i = \max\{0, F_i - d_i\}$ for each job i and then the total weighted flow time $F_w = \sum_{i=1}^{n} F_i(S) w_i$ for the sequence S. Job j is tardy if $T_j > 0$ and is non-tardy if $T_j = 0$. The number of tardy jobs will be denoted by $\#T$.

Probably the two most fundamental results in this context are:

Result 1: (Smith [5])

to minimize the total weighted flow time F_w of all jobs, schedule them in the non-decreasing order of p_i/w_i. If $S = \langle a1, a2, \ldots, an \rangle$ is such a schedule, then

$$\frac{p_{a1}}{w_{a1}} \leq \frac{p_{a2}}{w_{a2}} \leq \ldots \leq \frac{p_{an}}{w_{an}}$$

Result 2: (Jackson [4])

to minimize the maximum lateness or maximum tardiness ($T_{\max}$), jobs should be scheduled in the order of non-decreasing due-dates, the earliest due-date (EDD) schedule.

Throughout the remaining sections of this paper we will assume that the jobs are numbered in the non-decreasing order of due-dates so that the earliest due date (EDD) schedule ES is $\langle 1, 2, \ldots, n \rangle$. In the following section we will solve the one machine scheduling problem $n/1/F_w|T_{\max} = 0$, minimizing the sum of weighted flow times of all jobs with zero maximum tardiness. It will be assumed that there is at least one feasible schedule with $T_{\max} = 0$, that is: all jobs are early in the EDD schedule ES.

4. The $n/1/F_w|T_{\max} = 0$ Problem

The $n/1/F_w|T_{\max} = 0$ scheduling problem was first considered by Smith [5]. He conjectured that the optimum schedule for this problem can be obtained by scheduling job k last where $(p_k/w_k) = \max_{j \in L}\{p_j/w_j\}$ and $L = \{i \mid d_i \geq \sum_{r \in J} p_r, i \in J\}$ is the set of jobs that will not be tardy even if scheduled last. Even though Smith's conjecture for the $n/1/F_w|T_{\max} = 0$ problem seems to be a reasonable extension of result 1, Burns [2] and Emmons [3] have given counter examples. However the algorithm proposed by Burns [2] for this problem does not guarantee optimality. In this section we will present a branch and bound algorithm, incorporating decomposition principles to obtain the optimal schedule. Since condition $T_{max} = 0$ implies $\#T=0$, both $n/1/F_w|\#T = 0$ and $n/1/F_w|T_{max} = 0$ are equivalent and therefore this algorithm will also solve $n/1/F_w|\#T = 0$. As a first step we will show that the set J of jobs in $n/1/F_w|\#T = 0$ problem can be decomposed into subsets $J(i)$, $i=1,2,\ldots,m+1$, satisfying condition 1. Following this, two theorems will be formulated and proved in order to reduce much branching. A lower bound and the condition for sequence dominance will also be presented.

4.1 Decomposability of the $n/1/F_w|\#T = 0$ problem

The applicability of decomposition principle to $n/1/F_w|\#T = 0$ will be illustrated in this section. First we will present an algorithm that will provide us with the job partitions $J(i)$, $i=1,2,\ldots,m+1$, such that

$$\bigcup_{i=1}^{m+1} J(i) = J,$$

$$d_{t(k)} - \sum_{i=1}^{t(k)} p_i < \min_{i \in J(k-1)} \{p_i\}, \quad k=2,\ldots,m+1,$$

$$J(i) = \{t(i+1)+1, t(i+1)+2,\ldots,t(i)\}, \quad i=1,2,\ldots,m$$

and

$$J(m+1) = \{1,2,\ldots,t(m+1)\} .$$

That is, the earliness of job $t(k)$ in the EDD schedule is less than the minimum processing time of the jobs in the subset $J(k-1)$. Let ES = $\langle 1,2,\ldots,n \rangle$ be the EDD schedule.

Algorithm 4.1:

STEP 0: Initialize the sequence $S = \langle 1,2,\ldots,n \rangle$; Set $Q = J = \{1,2,\ldots,n\}$, $r = 1$, $\ell = n-1$, $t(1) = n$ and compute $F_\ell(S) = \sum_{i=1}^{\ell} p_i$. Go to step 1.

STEP 1: <u>If</u> $d_\ell - F_\ell(S) < \min_{k>\ell, k \in Q} \{p_k\}$, <u>then</u> go to step 2: <u>else</u> go to step 3.

STEP 2: Reset $S = \langle 1,2,\ldots,\ell \rangle$; $J(r) = Q - \{S\}$, where $\{S\}$ is the set of all jobs in the sequence S; $Q = \{S\}$; $r = r+1$; and $t(r) = \ell$. Go to step 3.

STEP 3: Set $\ell = \ell-1$. <u>If</u> $\ell = 0$, <u>then</u> set $m = r-1$; $J(m+1) = Q$; $t(m+2) = 0$ and stop; <u>else</u> compute $F_\ell(S) = \sum_{i=1}^{\ell} p_i$ and return to step 1.

Theorem 4.1. If $J(i)$, $i=1,2,\ldots,m+1$, are the subsets of jobs obtained by the above algorithm then all jobs $j\varepsilon J(i)$ should precede all jobs $k\varepsilon J(i-1)$ in a feasible (and therefore in an optimal) schedule $i=2,3,\ldots,m+1$.

Proof: Let us first examine the job partitions $J(i)$, $i=1,2,\ldots,m+1$, obtained through the above procedure. Since ℓ is the last job in a subset of jobs say $J(i)$ and since we denote this job by $t(i)=\ell$ in step 2, the subsets are

$$J(i) = \{t(i+1)+1,\ t(i+1)+2,\ldots,t(i)\},\quad i=1,2,\ldots,m,$$

and $$J(m+1) = \{1,2,\ldots,t(m+1)\}\ .$$

It should be noted that depending on the processing times and due-dates of the jobs the value of m can be $0 \leq m \leq n-1$. Obviously $m=0$ implies that the given set J could not be partitioned. In the following proof we will assume $m>0$ rejecting the trivial case $m=0$.

We will prove the theorem by showing that any arbitrary schedule R that does not satisfy the precedence constraints of theorem 4.1 will have at least one tardy job. Inspection of steps 1 and 2 of algorithm 4.1 will show that

$$\text{(1)}\qquad d_{t(k)} - \sum_{i=1}^{t(k)} p_i < \min_{i\varepsilon J(k-1)} \{p_i\},\ k=2,\ldots,m+1.$$

Let us consider an arbitrary schedule R of the n jobs, which may violate the precedence constraint of theorem 4.1.

Assume that all jobs in the set $\bigcup_{i=\ell}^{m+1} J(i) = \{1,2,\ldots,t(\ell)\}$ precede all jobs in $\bigcup_{i=1}^{\ell-1} J(i) = \{t(\ell)+1,\ldots,n\}$ in this schedule R for some ℓ. Let the last job to be processed among those in the subset $\bigcup_{i=\ell+1}^{m+1} J(i) = \{1,2,\ldots,t(\ell+1)\}$, in schedule R be job a. If job a follows at least one job $b\varepsilon J(\ell)$ in schedule R, then schedule R violates the precedence constraint of theorem 4.1. Also the flow time of job a is $F_a(R) \geq \sum_{i=1}^{t(\ell+1)} p_i + p_b$ because job a follows all the jobs in $\{1,2,\ldots,t(\ell+1)\}$ and b. Since the jobs are numbered in EDD order and $a \leq t(\ell+1)$ we have $d_a \leq d_{t(\ell+1)}$. Therefore

$$d_a - F_a(R) \leq d_{t(\ell+1)} - \sum_{i=1}^{t(\ell+1)} p_i - p_b.$$

However from (1) with $k=\ell+1$ and since $p_b \geq \min_{i\varepsilon J(\ell)} \{p_i\}$ because $b\varepsilon J(\ell)$ we have $d_a - F_a(R) < 0$. Therefore job a is tardy in the sequence R which implies that R is an infeasible schedule for the $n/1/F_w|\#T = 0$ scheduling problem. Therefore we can conclude that if $J(i)$, $i=1,2,\ldots,m+1$, are the subsets obtained by algorithm 4.1, then:

(2) if all jobs in the subset $\bigcup_{i=\ell}^{m+1} J(i)$ precede all jobs in $\bigcup_{i=1}^{\ell-1} J(i)$, then all jobs in the job set $\bigcup_{i=\ell+1}^{m+1} J(i)$ should precede all jobs in $\bigcup_{i=1}^{\ell} J(i)$

Note that the above result is true even when the set $\bigcup_{i=1}^{\ell-1} J(i)$ is empty. Therefore for $\ell=1$, and defining $\bigcup_{i=1}^{0} J(i) = \{\phi\}$ we have: all jobs in $\bigcup_{i=2}^{m+1} J(i)$ should precede all jobs in $J(1)$, in a feasible schedule. Then by induction from (2) we have: all $j\varepsilon \bigcup_{i=\ell}^{m+1} J(i)$ should precede all $k\varepsilon \bigcup_{i=1}^{\ell-1} J(i)$ in a feasible schedule for all $\ell=2,3,\ldots,m+1$. Therefore all $j\varepsilon J(i)$ should precede all $k\varepsilon J(i-1)$ in a feasible schedule for all $i=2,\ldots,m+1$.

Q.E.D.

Note that if $m=n-1$, the EDD schedule is the only feasible schedule and therefore it is optimal.

Now we will define the subproblems $P(i)$ on $J(i)$, $i=1,2,\ldots,m+1$, such that the objective function $f_i(\cdot)$ for the subproblem of jobs $J(i)$, $i=1,2,\ldots,m+1$, satisfy condition 1.2.

Definition 4.1 (Subproblem)

If $J(i)$ with cardinality $|J(i)| = n_i$, $i=1,2,\ldots,m+1$, were obtained by algorithm 4.1, then define the subproblem $P(k)$, $\forall\ k$ ($k=1$ to $m+1$) as follows: minimize the weighted sum of flow time of all n_k jobs in the set $J(k)$ with zero maximum tardiness, processing times p_i, due dates d_i' and weights w_i, $i\varepsilon J(k)$, where

$$d_i' = \begin{cases} d_i & i\varepsilon J(m+1) \\ d_i - \sum_{j=1}^{t(k+1)} p_j & i\varepsilon J(k),\ k<m+1 \end{cases}$$

That is, if we define the original problem P as $\{J;\ p_i,\ d_i,\ w_i;\ n/1/F_w|\#T = 0\}$, then the subproblems are $\{J(k);\ p_i,\ d'_i,\ w_i;\ n/1/F_w|\#T = 0\}$, $k=1,2,\ldots,m+1$ where the objective function $f_k=F_w$.

Let $\pi_k = \langle\pi_k(i),\ i=1,2,\ldots,n_k\rangle$ be the schedule of all jobs in $J(k)$ and let $F_i(\pi_k)$ be the flow time of job $i\varepsilon J(k)$ for the sequence π_k for the problem $P(k)$. Then

$$f_k(\pi_k) = \sum_{i\varepsilon J(k)} F_i(\pi_k)w_i = \sum_{i=1}^{n_k} w_{\pi_k(i)} \sum_{j=1}^{i} p_{\pi_k(j)} .$$

Now we will define $h(\cdot)$ given in condition 1.2 by the following theorem.

Theorem 4.2. Let $S = \langle\pi_{m+1},\ \pi_m,\ldots,\pi_1\rangle$ be an arbitrary schedule of the n jobs in J, while π_k is the schedule of jobs in $J(k)$, $k=1,2,\ldots,m+1$. The total cost of schedule S in P is $\phi(S) = \sum_{i\varepsilon J} F_i(S)w_i$. If we define

$$h(f_k(\pi_k),\ k=1,2,\ldots,m+1) = \sum_{k=1}^{m+1} (f_k(\pi_k) + R_k \sum_{i\varepsilon J(k)} w_i) ,$$

where $R_k = \sum_{i=1}^{t(k+1)} p_i$, then $\phi(S) = h(f_k(\pi_k),\ k=1,2,\ldots,m+1)$.

Proof: Rewriting the objective function of P for the sequence S, we have

$$(3)\qquad \phi(S) = \sum_{i\varepsilon J} F_i(S)w_i = \sum_{k=1}^{m+1} \left(\sum_{i\varepsilon J(k)} (F_i(S)w_i)\right)$$

Since all jobs in $\bigcup_{i=k+1}^{m+1} J(i)$ precede all jobs in the subset $J(k)$ in S (and always in a feasible schedule),

$$F_{\pi_k(i)}(S) = \sum_{j=1}^{t(k+1)} p_j + \sum_{j=1}^{i} p_{\pi_k(j)}, \qquad \pi_k(i)\ \varepsilon\ J(k)$$

Therefore

$$(4)\qquad \sum_{i\varepsilon J(k)} F_i(S)w_i = \sum_{j=1}^{t(k+1)} p_j \sum_{i\varepsilon J(k)} w_i + \sum_{i=1}^{n_k} (w_{\pi_k(i)} \sum_{j=1}^{i} p_{\pi_k(j)})$$

$$= R_k \sum_{i\varepsilon J(k)} w_i + f_k(\pi_k)$$

Then from (3), (4) and the definition of $h(\cdot)$, we get

$$(5)\qquad \phi(S) = h(f_k(\pi_k),\ k=1,2,\ldots,m+1)$$

Note that $\sum_{k=1}^{m+1} R_k \sum_{i \in J(k)} w_i$ is independent of the sequences π_k, $k=1,2,\ldots,m+1$, and therefore the function $h(\cdot)$ is monotonically non-decreasing on the vector $\mathcal{F} = (f_1, f_2, \ldots, f_{m+1})$.

Q.E.D.

In order to complete the proof for the validity of the decomposition of problem P, we have to show that the condition $\#T=0$ will not be violated by the schedule $S = \langle\pi_{m+1}, \pi_m, \ldots, \pi_1\rangle$ when π_k is a feasible schedule to the sub-problem $P(k)$, $k=1,2,\ldots,m+1$. To show this let us choose an arbitrary job j from a job set say $J(k)$. Let π_k be a feasible schedule to $P(k)$. Then

$$F_j(\pi_k) \leq d'_j = d_j - \sum_{i=1}^{t(k+1)} p_i$$

Therefore

$$d_j \geq F_j(\pi_k) + \sum_{i=1}^{t(k+1)} p_i \tag{6}$$

$$= F_j(S)$$

Hence job j is early in schedule S. Q.E.D.

Therefore it follows that the subproblems formed according to definition 4.1 satisfy condition 1 for complete decomposition. In the remainder of this section we will consider the properties of the $n/1/F_w|\#T = 0$ problem and propose an algorithm that will utilize these properties and the decomposition principle and give the optimal schedule.

4.2. Job exclusion and job dominance

The counter examples given by Burns [2] and Emmons [3], to Smith's conjecture demonstrate the fact that more than one job may be suitable to be scheduled last when using the principle of Smith's conjecture. That is: let Q be the set of jobs to be scheduled and $L = \{i | d_i \geq \sum_{r \in Q} p_r,\ i \in Q\}$, then all the jobs in the set L may be suitable to be scheduled last. Brucker, Lenstra and Rinnooy Kan [1] have shown that $n/1/F_w|\#T = 0$ is NP complete. This fact indeed eliminates the possibility of obtaining a simple search algorithm. In this regard we will employ a branch and bound approach to solve this one machine scheduling problem. The expansion of a node will be carried out by choosing the jobs that could be placed last among those jobs to be scheduled. Let Q_a be the set of jobs to be scheduled at node a and the jobs in the set $J-Q_a$ have been already scheduled in the order σ_a. If $L_a = \{i | d_i \geq \sum_{r \in Q_a} p_r,$

$i \varepsilon Q_a\}$ and $|L_a| = m_a$, then m_a additional nodes can be generated by forming partial sequence $<i, \sigma_a>$ for each job $i \varepsilon L_a$. The nodes will be numbered in the order they are formed and the node with the largest number in the node list will be selected next for expansion.

Now we will analyse the possibilities of eliminating unnecessary branches that we are sure will not lead to an optimal schedule. The following theorem provides a sufficient condition to eliminate jobs that give rise to such branches.

Theorem 4.3. (Job exclusion) Let Q_a be the set of jobs to be scheduled, σ_a be the partial schedule created up to node a and $L_a = \{i | d_i \geq \sum_{r \varepsilon Q_a} p_r,$ $i \varepsilon Q_a\}$. Then $J = Q_a \cup \{\sigma_a\}$ and L_a is the set of jobs potentially available to be scheduled last, where $\{\sigma_a\}$ is the set of jobs in σ_a. Then job $j \varepsilon L_a$ need not be scheduled last if $w_j \geq w_k$ and $p_j \leq p_k$ for some $k \varepsilon L_a$. That is job k excludes j from L_a (k EX j). Here and hereafter it will be assumed that ties are broken arbitrarily.

Proof: Let $S = <\rho, \sigma_a>$ be an optimal schedule obtained from the development of node a. $\rho = <\rho 1, \rho 2, \ldots, k, \ldots, j>$, $w_j \geq w_k$ and $p_j \leq p_k$. Note that the jobs in the schedule ρ forms the set $\{\rho\} = Q_a$. Suppose $S' = <\rho', \sigma_a>$ is an alternate schedule obtained by interchanging jobs j and k in sequence S so that $\rho' = <\rho 1, \rho 2, \ldots, j, \ldots, k>$. Since $p_j \leq p_k$, the flow times $F_i(S)$ and $F_i(S')$ in the sequences S and S' satisfy $F_i(S') \leq F_i(S)$, $i \neq j,k$, $F_k(S') = F_j(S)$ and $F_j(S') = F_k(S) - p_k + p_j \leq F_k(S)$. Therefore

$$\sum_{i \varepsilon J} F_i(S')w_i - \sum_{i \varepsilon J} F_i(S)w_i \leq F_j(S')w_j + F_k(S')w_k - F_j(S)w_j$$
$$- F_k(S)w_k \leq (F_j(S) - F_k(S))(w_k - w_j)$$

Since $w_k \leq w_j$ and $F_j(S) > F_k(S)$ we have

$$\sum_{i \varepsilon J} F_i(S')w_i - \sum_{i \varepsilon J} F_i(S)w_i \leq 0$$

Therefore S' is an alternate optimal schedule. Hence schedule S need not be considered.

Q.E.D.

Let L'_a be the set of jobs obtained by excluding all such jobs j from L_a. That is for each job j in the set $L_a - L'_a$, there is at least one job k in L'_a such that $w_j \geq w_k$ and $p_j \leq p_k$. Hereafter the representation of L'_a will be interpreted as defined here. In order to eliminate unnecessary search for non-optimal schedules, we will provide some useful theorems. For this we will first define 'job dominance'.

Definition 4.2

Job j dominates job k (jDk) if $(p_j/w_j) \leq (p_k/w_k)$, ties being broken arbitrarily.

Now let Q_b be the set of jobs to be scheduled and H_b be the set of jobs potentially available to be scheduled in the last position at node b. Assume that the jobs in the set H_b are numbered such that $H_b = \{j1, j2, \ldots, jh\}$, where $(p_{ji}/w_{ji}) \leq (p_{j(i+1)}/w_{j(i+1)})$, $i=1,2,\ldots,h-1$, and $h = |H_b|$ is the number of jobs in H_b. Then for each job ji, develop a node $b+i$ such that

$$Q_{b+i} = Q_b - \{ji\}; \quad \sigma_{b+i} = \langle ji, \sigma_b \rangle$$

and $D_{b+i} = \{k \mid ji \text{ D} k, k \epsilon H_b\}$ is the set of jobs dominated by ji. Now we will show that the jobs in the set D_{b+i} at node $b+i$ need not be considered for the last position among the jobs in Q_{b+i}.

Theorem 4.4. If Q_a is the set of jobs to be scheduled at node a and D_a is the set of jobs obtained as discussed above, then jobs $j \epsilon D_a$ need not be considered for the last position among the jobs in Q_a.

Proof: Let us assume that node a is generated from node b by assigning job k so that $\sigma_a = \langle k, \sigma_b \rangle$. If $j \epsilon D_a$, then kDj and $j \epsilon H_b$. Let $S = \langle \rho, j, k, \sigma_b \rangle$ be an optimal schedule obtained from the development of node a (see Figure 1). Now consider the schedule $S' = \langle \rho, k, j, \sigma_b \rangle$. Since $j \epsilon H_b$, S' is

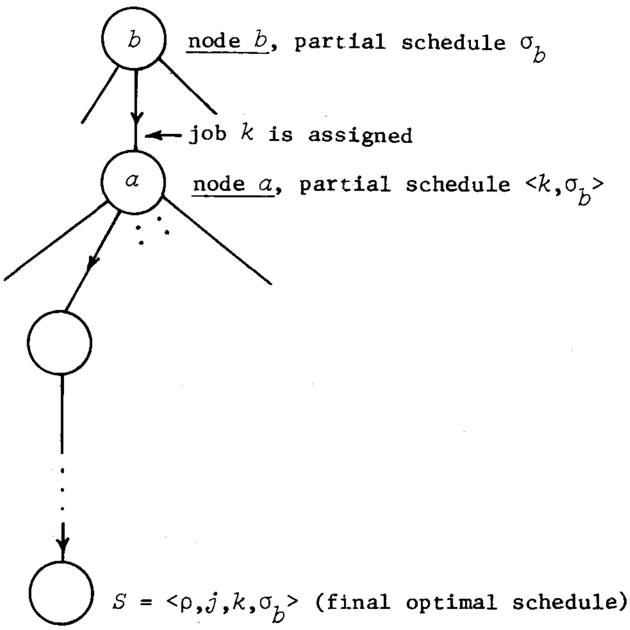


Figure 1.

feasible (since H_b is the set of jobs potentially available to be scheduled at nobe b) and since $k D j$, from result 1 we know that S' is an alternate optimal schedule. Therefore we need not consider sequences in the form of S.

Q.E.D.

Hence we can reduce the set of potentially available jobs in L'_a as follows: The jobs that should be considered for the last position among the jobs in Q_a at node a are in the set H_a, where $H_a = L'_a - L'_a \cap D_a$ and the set D_1 at the first node should be assumed to be empty. That is $H_1 = L'_1$.

We will now consider two more possibilities of eliminating unnecessary nodes. This will be accomplished by:

(1) defining a lower bound,

and (2) defining a condition for sequence dominance.

4.3. Lower bound

Let Q_a be the set of jobs to be scheduled and σ_a be the already scheduled partial sequence. Then we would like to obtain a lower bound $LB_a \leq \min_{\rho\varepsilon\Sigma} \{ \sum_{i\varepsilon J} F_i(S)w_i \}$, where $S = <\rho, \sigma_a>$ and Σ is the set of all possible schedules of jobs in Q_a that do not violate the non tardy condition. Then if $P\Sigma$ is the set of all permutation schedules of jobs in Q_a, we have $\Sigma \subset P\Sigma$ and therefore,

$$\min_{\rho\varepsilon\Sigma} \{ \sum_{i\varepsilon J} F_i(S)w_i \} \geq \min_{\rho\varepsilon P\Sigma} \{ \sum_{i\varepsilon J} F_i(S)w_i \} ,$$

where $S = <\rho, \sigma_a>$. If ρ^* is the non-decreasing order of (p_i/w_i) of all jobs in Q_a and $S^*_a = <\rho^*, \sigma_a>$, then from result 1, we have

$$\min_{\rho\varepsilon P\Sigma} \{ \sum_{i\varepsilon J} F_i(S)w_i \} = \sum_{i\varepsilon J} F_i(S^*_a)w_i .$$

Then we can choose $LB_a = \sum_{i\varepsilon J} F_i(S^*_a)w_i$ as our lower bound.

4.4. Sequence dominance

Let σ_a and σ_b be two already scheduled partial sequences respectively at nodes a and b such that the jobs in them are the same. That is $\{\sigma_a\} = \{\sigma_b\}$. Also let $C(\sigma_r) = \sum_{i\varepsilon\{\sigma_r\}} F_i(\rho\sigma_r)w_i$ for $r=a$ and b, for an arbitrary sequence ρ of jobs in Q_a ($\overline{=}$ Q_b) where the schedules $\rho\sigma_r = <\rho, \sigma_r>$ for $r=a$ and b. Note that $C(\sigma_r)$ is independent of ρ for $r=a,b$.

Theorem 4.5. If $C(\sigma_a) \geq C(\sigma_b)$, then the partial sequence σ_a can be dropped from further expension.

Proof: The weighted sum of flow times for the sequence $\rho\sigma_a$ is given by,

$$F_w(\rho\sigma_a) = \sum_{i\varepsilon J} F_i(\rho\sigma_a)w_i$$
$$= \sum_{i\varepsilon Q_a} F_i(\rho)w_i + \sum_{i\varepsilon\{\sigma_a\}} F_i(\rho\sigma_a)w_i$$
$$\geq F_w(\rho\sigma_b), \qquad \forall\ \rho,$$

because $C(\sigma_a) \geq C(\sigma_b)$. Q.E.D.

We will now present the algorithm to solve $n/1/F_w|\#T = 0$. We will note again that the jobs are numbered in the EDD schedule order. The upper bound UB will be the weighted flow time of the best known schedule. UB will be set equal to ∞ when no feasible schedule is known and updated every time a schedule with a lower weighted flow time becomes known.

4.5. The algorithm

Algorithm 4.2 (for $n/1/F_w|\#T = 0$)

STEP 1 : DECOMPOSITION: Obtain the job partitions $J(k)$, k=1,2,...,m+1, using algorithm 4.1. Initialize r=1, reset $d_i = d_i - \sum_{j=1}^{t(k+1)} p_j$, $i\varepsilon J(k)$, k=1,2,...,m, and define the subproblem $P(k)$ as given by definition 4.1. That is $P(k) = \{J(k),\ p_i,\ w_i,\ d_i;\ n_k/1/F_w|\#T = 0\}$, where $n_k = |J(k)|$, k=1,2,...,m+1. Go to step 2.

STEP 2 : Choose subproblem $P(r)$ and go to step 2.0.

STEP 2.0: Initialize $\sigma_1 = D_1 = \{\phi\}$ empty; node entry $N = \{1\}$; $Q_1 = J(r)$ and $UB = \infty$. Go to step 2.1.

STEP 2.1: <u>If</u> N is empty, <u>then</u> go to step 2.5; <u>else</u> choose the node (say a) with the highest node number and eliminate it from the node list. Obtain the lower bound LB_a as discussed in section 4.3. <u>If</u> $LB_a \geq UB$, <u>then</u> reject node a and repeat step 2.1: <u>else</u>, if $|Q_a| = 1$, then estimate $f_r(Q_a,\ \sigma_a) = \sum_{i\varepsilon J(r)} F_i w_i$ for the sequence $<Q_a,\ \sigma_a>$. If $f_r(Q_a\ \sigma_a) < UB$, then set $UB = f_r(Q_a\ \sigma_a)$ and $\pi_r^* = <Q_a,\ \sigma_a>$ and repeat step 2.1: else reject this schedule and repeat step 2.1: else go to step 2.2: <u>else</u> go to step 2.2.

STEP 2.2: Obtain $L_a = \{i|\ d_i \geq \sum_{j\varepsilon Q_a} p_j,\ i\varepsilon Q_a\}$. <u>If</u> $L_a = Q_a$, <u>then</u> arrange the jobs in Q_a in the non-decreasing order of the ratio (p_j/w_j) and let this schedule be ρ_a. Now define $\pi_a = <\rho_a,\ \sigma_a>$. If $f_r(\pi_a) < UB$, then update the upper bound solution by setting $UB = f_r(\pi_r)$ and $\pi_r^* = \pi_a$. Go to step 2.1: else reject this schedule and go to step 2.1: else using Theorem 4.4 find the set L_a' such that $j \varepsilon L_a - L_a'$, there is at least one $k \varepsilon L_a'$ such that $w_j \geq w_k$ and $p_j \leq p_k$. Find $H_a = L_a' - L_a' \cap D_a$ and go to step 2.3.

STEP 2.3: Let $H_a = \{j1, j2, \ldots, jh\}$ and $h = |H_a|$ such that $(p_{ji}/w_{ji}) \leq (p_{j(i+1)}/w_{j(i+1)})$. Add to the node list a new node $a+i$ for each job $ji \varepsilon H_a$, $i=1,2,\ldots,h$, numbering them $a+1, a+2, \ldots, a+h$, with $\sigma_{a+i} = \langle ji, \sigma_a \rangle$, $Q_{a+i} = Q_a - \{ji\}$, $D_{a+i} = \{k \mid ji \mathrm{D} k, k \varepsilon H_a\}$. Then the new node list $N = N \cup \{a+1, \ldots, a+h\}$. Go to step 2.4.

STEP 2.4: Check for sequence dominance (as in Theorem 4.5) for all partial schedules corresponding to the nodes in N. Eliminate the nodes with dominated schedules from the node list and go to step 2.1.

STEP 2.5: π_r^* is the optimal schedule for subproblem $P(r)$ and $f_r(\pi_r^*) = UB$. Go to step 3.

STEP 3 : If $r=m+1$, then go to step 4: else set $r=r+1$ and go to step 2.

STEP 4 : Obtain $S^* = \bigcup_{i=0}^{m} \pi^*_{m+1-i} = \langle \pi^*_{m+1}, \ldots, \pi^*_1 \rangle$ and $\phi(S^*) = \sum_{k=1}^{m+1} (f_k(\pi_k^*) + R_k \sum_{i \varepsilon J(k)} w_i)$, where $R_k = \sum_{i=1}^{t(k+1)} p_i$. Then S^* is the optimal schedule and $\phi(S^*)$ is the minimum weighted flow time with zero maximum tardiness. Stop.

EXAMPLE:

As an example consider the $10/1//F_w|\#T = 0$ problem;

Table 1

job i	1	2	3	4	5	6	7	8	9	10
p_i	3	8	7	9	4	6	13	10	7	4
ω_i	3	6	7	2	3	9	5	4	2	6
d_i	15	19	27	33	34	57	59	72	73	79

Table 1 gives the data matrix and Table 2 gives the flow times of the jobs according to the EDD schedules, and the set partitioning point (job 5).

Table 2

job i	1	2	3	4	5	6	7	8	9	10
$F_i(S)$	3	11	18	27	31	37	50	60	67	71
d_i	15	19	27	33	34	57	59	72	73	79
$d_i-F_i(S)$	12	8	9	6	3	20	9	12	6	8
$\mathrm{Min}_{j>i}(p_j)$	4	4	4	4	4	4	4	4	4	4
$t(r)$					5					
r					1					

There are two subsets (i.e. m=1),

$J(1)$ = 6,7,8,9,10 and

$J(2)$ = 1,2,3,4,5

Then the subproblem $P(1)$ is given by the following data matrix.

$P(1)$: $5/1//F_w|\#T = 0$, $R_1 = 31$

job i	6	7	8	9	10
p_i	6	13	10	7	4
ω_i	9	5	4	2	6
d_i	26	28	41	42	48

Similarly subproblem $P(2)$: $5/1//F_w|\#T = 0$; $R_2 = 0$

job i	1	2	3	4	5
p_i	3	8	7	9	4
ω_i	3	6	7	2	3
d_i	15	19	27	33	34

Consider the subproblem $P(2)$:

Selected node:

⇩			N ⇩
	$L = \sigma_1 = \phi$	Node List	{1}
$\eta = 1$;	$L = \{4,5\}$; 4 Ex 5 ;	$\sigma_2 = 4$	{2}
$\eta = 2$;	$L = \{3,5\}$; 3 D 5 ;	$\sigma_3 = 3,4$	
		$\sigma_4 = 5,4$	{3,4}

$\eta = 3$; $L = \{1,2,5\} = \tilde{\pi}_3$: Since all jobs to be sequenced belong to the set L; π_3=<1,5,2> and π_2^* = <1,5,2,3,4> and UB = 336.

$\eta = 4$; $L = \{2,3\}$; 2 Ex 3 ; σ_5 = <2,5,4> {5}

$\eta = 5$; $L = \{1,3\} = \tilde{\pi}_5$; $\pi_5\sigma_5$=<1,3,5,2,4> and $\phi(\tilde{\pi}_5\sigma_5) = 315$

< 336. Therefore $\pi_2^* = \langle 1,3,5,2,4 \rangle$ and $\phi(\pi_2^*) = 315$.

Similarly, for the subproblem $P(1)$, the sequence $\pi_1^* = \langle 10,6,7,8,9 \rangle$ was obtained, with $\phi(\pi_1^*) = 445$.

Then $\pi^* = \pi_2^* \pi_1^* = \langle 1,3,5,2,4,10,6,7,8,9 \rangle$ and

$$\phi(\pi^*) = 31 \times (26) + 315 + 445 = 1566.$$

The resulting tree structures for the subproblems $P(1)$ and $P(2)$ are given in Figures 2 and 3 respectively.

SUBSET 1:

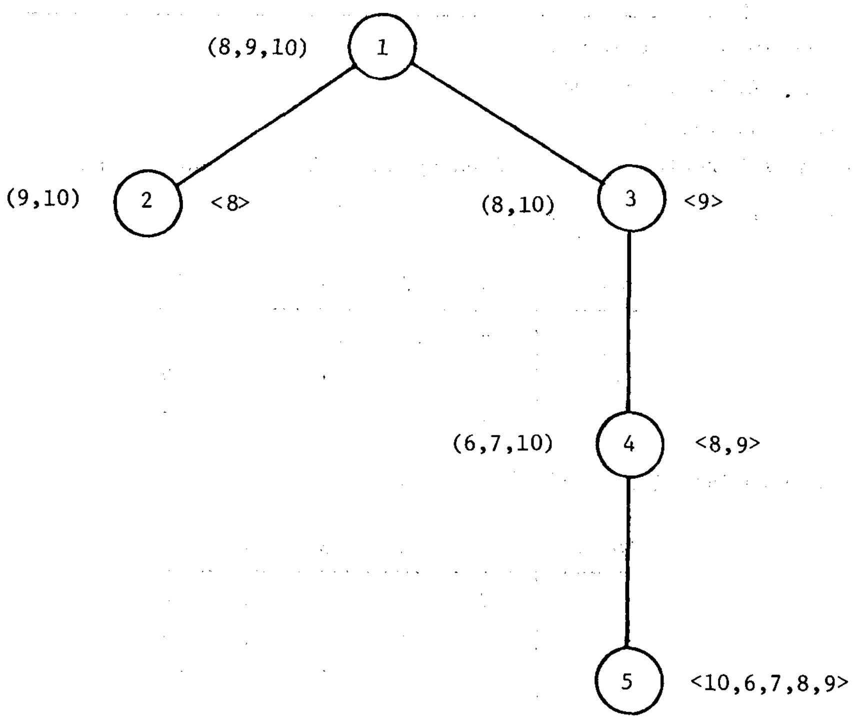
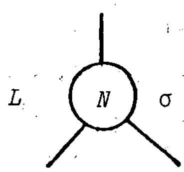


Figure 2.

Nomenclature:



L: the set L at each node level is given within the bracket, left to the node circle.

σ: the scheduled partial sequence is shown at the right of the node circle.

N: node numbers are marked within the node circles.

SUBSET 2:

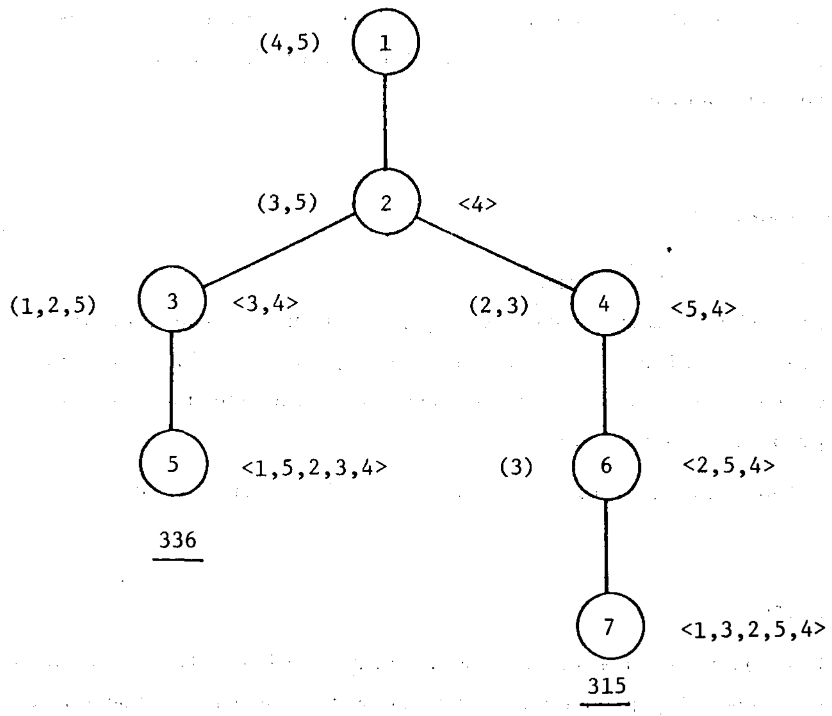


Figure 3.

In the next section we will extend the above algorithm to obtain the optimal schedule to $n/1/F_w|T_{max}$. For this we will first show that $n/1/F_w|T_{max}$ is reducible to $n/1/F_w|\#T = 0$.

5. The $n/1/F_w|T_{max}$ Problem

Let $T_{max} = \max_{i\varepsilon J} \{T_i\}$ be the maximum tardiness in the EDD schedule. In $n/1/F_w|T_{max}$, it is required to minimize the weighted sum of flow times while the maximum tardiness is kept equal to T_{max}. Let this problem be P = $\{J;\ p_i,\ w_i,\ d_i;\ n/1/F_w|T_{max}\}$ and define $P' = \{J;\ p_i,\ w_i,\ d_i';\ n/1/F_w|\#T = 0\}$, where

$d_i' = d_i + T_{max}, \ \forall \ i \in J.$

5.1. Reducibility of P to P'

In this section we will show that P can be reduced to P'.

Theorem 5.1. The problem P can be reduced to P'.

Proof: Since $d_i' = d_i + T_{max}$, it is easily verified that all feasible schedules to P are also feasible to P'. Since the processing times and weights are the same in both P and P', the weighted sum of flow times in P and P' are the same for any schedule.

Q.E.D.

5.2. The algorithm

ALGORITHM 5.1 (for $n/1/F_w|T_{max}$)

STEP 1: Find the maximum tardiness T_{max}, by arranging the jobs in the EDD order. Define a new problem $P' = \{J;\ p_i,\ w_i,\ d_i';\ n/1/F_w|\#T = 0\}$, where $d_i' = d_i + T_{max}, \ \forall \ i \in J$ and d_i is the due date of the original problem. Go to step 2.

STEP 2: Solve P' using Algorithm 4.2. S^* is the optimal schedule. Stop.

6. Conclusions

In this paper we have given a sufficient condition to decompose a large permutation scheduling problem into two or more scheduling problems of smaller size. We have also demonstrated the applicability of this decomposition approach to scheduling problems by incorporating it in the branch and bound algorithm to solve $n/1/F_w|\#T = 0$ and $n/1/F_w|T_{max}$. Since these two scheduling problems can be completely decomposed the effectiveness of the proposed algorithms is evident.

Acknowledgements

The authors would like to thank the referees for their thorough reading and their useful comments which helped to improve the clarity of the presentation of this paper. The financial support for this research for the first author was provided by the Canadian Commonwealth Scholarship and Fellowship Committee.

References

[1] Brucker, P., J. K. Lenstra and A. H. J. Rinnooy Kan: Complexity of Machine Scheduling Problems, *Report BW 43/75*, Mathematisch Centrum, Amsterdam, 1975.

[2] Burns, R. N.: Scheduling to Minimize the Weighted Sum of Completion Times with Secondary Criteria, *Naval Research Logistics Quarterly*, Vol. 23 (March 1976) 125-130.

[3] Emmons, H.: A Note on a Scheduling Problem with Dual Criteria, *Naval Research Logistics Quarterly*, Vol. 22 (Sept. 1975) 615-616.

[4] Jackson, J. R.: Scheduling a Production Line to Minimize Maximum Tardiness, *Research Report 43*, Management Science Research Project, UCLA, (January 1955).

[5] Smith, W. E.: Various Optimizers for Single Stage Production, *Naval Research Logistics Quarterly*, Vol. 3 (March 1956) 59-66.

J. G. SHANTHIKUMAR: Assistant Professor
Dept. of IE & OR, 441 Link Hall
Syracuse University, Syracuse,
New York 13210, U.S.A.