The Pennsylvania State University

Computer Science Department
University Park, PA. 16802

Technical Report No. 213
December 1976

SCHEDULING INDEPENDENT TASKS WITH RELEASE DATES AND DUE DATES ON PARALLEL MACHINES

John Bruno and Teofilo Gonzalez

Introduction

An instance I of the scheduling problem considered in this paper is given by:

Two integers $n > 0$ and $m > 0$ and three sequences of numbers

$\tau_1, \tau_2, \ldots, \tau_n$

$r_1, r_2, \ldots, r_n$

and

$d_1, d_2, \ldots, d_n$

such that $\tau_i > 0$, $r_i \geq 0$, $d_i \geq 0$ and $d_i \geq r_i + \tau_i$ for $i = 1, \ldots, n$.

Informally, n is the number of tasks, m is the number of (identical) machines, the τ_i's are service times, the r_i's are release dates and the d_i's are due dates. The i^{th} task will be denoted by T_i. Task T_i requires a total of τ_i units of service. This service cannot begin before time r_i and may not extend beyond time d_i. This service need not be obtained from a single machine, that is, preemptions are allowed.

A service assignment is a tuple $\langle j,s,f \rangle$ where j is a positive integer less than or equal to m and s and f are nonnegative numbers such that $s < f$.

A schedule σ is a mapping from tasks into sets of service assignments. For example, $\sigma(T_i) = \{\langle j_1,s_1,f_1 \rangle, \langle j_2,s_2,f_2 \rangle\}$ means that machine j_1 services task T_i during the time interval $[s_1,f_1)$ and machine j_2 services task T_i during the time interval $[s_2,f_2)$. The total service given to task T_i is $(f_1 - s_1) + (f_2 - s_2)$.

We say a schedule σ is feasible if it satisfies the following conditions:

1. No machine is assigned so as to be servicing more than one task at any instant.
2. No task is receiving service from more than one machine at any instant.

3. No task is receiving service from any machine at some instant earlier than its release date or later than its due date.

4. The total service given to each task is equal to its service time.

In the next section we shall give an efficient algorithm for determining a feasible schedule σ (if one exists) given an instance I of our scheduling problem.

An Algorithm

Let I be an instance of our scheduling problem. Let $a_1 \le a_2 \le a_3 \le \dots \le a_{2n}$ be the ordered collection of release dates and due dates. For example, if $n = 3$,

$$r_1, r_2, r_3 = 1,1,3,$$

and

$$d_1, d_2, d_3 = 2,7,5$$

then

$$a_1, a_2, a_3, a_4, a_5, a_6 = 1,1,2,3,5,7.$$

Let $R_j = [a_j, a_{j+1})$ and $k_j = a_{j+1} - a_j$ for $j = 1,\dots,2n-1$. Some of the intervals R_j may be empty ($k_j = 0$) and can be discarded. Renumber the remaining intervals such that $R_1,\dots,R_p$ are all the nonempty intervals ($k_j > 0$).

We form a capacitated flow network $N(I)$ with vertices So (source), Si (sink), $T_1,\dots,T_n$ and $R_1,\dots,R_p$. The arcs in $N(I)$ and their capacities are:

arc	capacity	
(So, T_i)	τ_i	$i=1,\dots,n$
(R_j, Si)	mk_j	$j=1,\dots,p$
(T_i, R_j)	k_j	for all T_i and R_j such that T_i can be serviced in the interval R_j

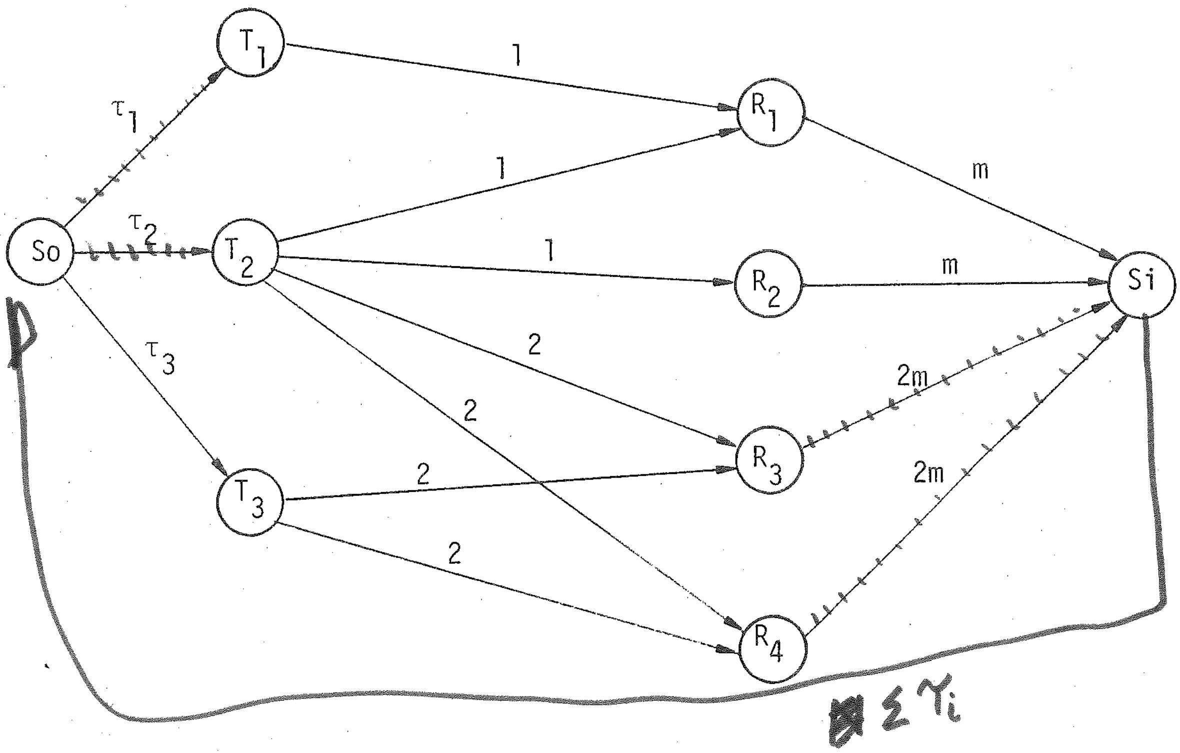
Continuing the above example we find (after eliminating interval [1,1)

$R_1 = [1,2)$ $k_1 = 1$

$R_2 = [2,3)$ $k_2 = 1$

$R_3 = [3,5)$ $k_3 = 2$

$R_4 = [5,7)$ $k_4 = 2$



Flow Network

Let $c(A,B)$ denote the capacity of arc (A,B) in $N(I)$. Our objective is to determine a maximum flow from So to Si in $N(I)$. A flow pattern h is a mapping from arcs in $N(I)$ into nonnegative numbers such that $h(A,B) \leq c(A,B)$ for every arc (A,B) in $N(I)$, $F = \sum_{i=1}^{n} h(So,T_i) = \sum_{j=1}^{p} h(R_j,Si)$, and the sum of the flow into vertex T_i (R_j) is equal to the flow leaving vertex T_i (R_j) for $i=1,\ldots,n$ $(j=1,\ldots,p)$. The value of the flow pattern h is equal to F.

Algorithm A

Input: Instance I of scheduling problem

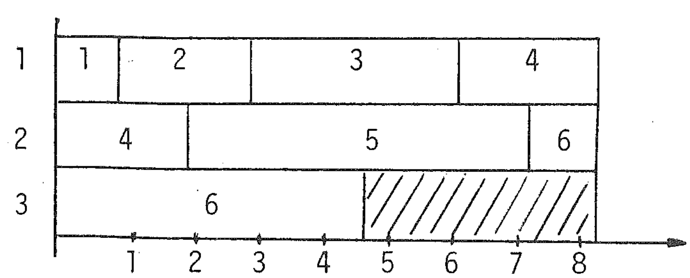
Output: Feasible schedule σ if one exists.

Method:

1. Construct flow network $N(I)$
2. Find flow pattern h with maximum value F.
3. If $F < \sum_{i=1}^{n} \tau_i$ then no feasible schedule exists; stop.
4. If $F = \sum_{i=1}^{n} \tau_i$ then use h to construct a feasible schedule σ. □

The construction of a feasible schedule in step 4 of Algorithm A is most easily accomplished one interval R_j at a time. Let $t_i = h(T_i, R_j)$ for each arc (T_i, R_j) (j is fixed) in $N(I)$ and $t_i = 0$ if (T_i, R_j) is not in $N(I)$. By the construction of $N(I)$ we have that $t_i \le k_j$ and $\sum_{i=1}^{n} t_i \le mk_j$. The above conditions are sufficient to enable us to determine service assignments to tasks T_i with $t_i > 0$ in the interval R_j. The algorithm will not be stated formally here but can be found in [Mc]. An example should suffice.

Let $t_1 = 1$, $t_2 = 2$, $t_3 = 3$, $t_4 = 4$, $t_5 = 5$, $t_6 = 6$, $R_1 = [0,8)$ and $m = 3$.



We begin by making service assignments on machine 1 for t_1, t_2 and t_3 time units. Next we see that t_4 won't "fit" on machine 1. We use up the remaining capacity of machine 1 by servicing task T_4 for two units and then use machine 2 for the remaining service. (Actually, the service from machine 2 occurs chronologically before the service from machine 1.) Since $t_4 \leq 8$ we are guaranteed that this service assignment satisfies condition 2 for feasibility.

In step 4 of Algorithm A we use h to construct σ in each of the intervals R_j for $j=1,\ldots,p$.

The correctness of Algorithm A rests on the following lemma.

Lemma 1 A feasible schedule exists for an instance I of our scheduling problem if and only if the maximum value F of a flow pattern for $N(I)$ is equal to $\sum_{i=1}^{n} \tau_i$.

Complexity of the Algorithm

In this section we argue that the worst-case time complexity of Algorithm A is $O(n^3)$. It takes time $O(n \log_2 n)$ to construct the R_j's, $O(n^2)$ to build $N(I)$ and, from Edmonds and Karp [EK], $O(n^3)$ to find F. The last step in the construction of σ is $O(n^2)$.

Uniform Machines

In this section we extend our results to the case of two uniform machines. Let m, the number of machines, be equal to 2. Suppose machine 1 has speed s_1 and machine 2 has speed s_2 where $s_1 \geq s_2$. Accordingly, it takes machine j ($j=1$ or 2) τ_i/s_j time units to service task T_i where τ_i is the service requirement of T_i. Suppose task T_i receives an actual service time of δ_j on machine j for $j=1,2$. We require that these total actual service times be such that $\delta_1 s_1 + \delta_2 s_2 = \tau_i$, the service requirement of task T_i.

As we did earlier, we define a capacitated flow network $\bar{N}(I)$ with vertices So (source), Si(sink), $T_1,\ldots,T_n$ and $R_1,\ldots,R_p$. The arcs in

$\bar{N}(I)$ and their capacities are:

arc	capacity	
(So, T_i)	τ_i	$i=1,\ldots,n$
(R_j, Si)	$(s_1+s_2)k_j$	$j=1,\ldots,n$
(T_i, R_j)	$s_1 k_j$	for all T_i and k_j such that T_i can be serviced in the interval R_j.

We have the following lemma.

Lemma 2 Let $m=2$ and $s_1 \geq s_2$. A feasible schedule exists for and instance I of our scheduling problem if and only if the maximum value $\bar{\bar{F}}$ of a flow pattern for $\bar{N}(I)$ is equal to $\sum_{i=1}^{n} \tau_i$.

Proof: Let σ be a feasible schedule. Let R be some region and let k be the length of R. Let δ_{ij} be the total actual service time of task T_i on machine j in region R.

Define $t_i = s_1\delta_{i1} + s_2\delta_{i2}$. Clearly, $\delta_{i1} + \delta_{i2} \leq k$ and therefore $t_i \leq s_1 k$ (we have used $s_1 \geq s_2$). Thus a flow of t_i units in the arc (T_i, R) does not exceed the capacity of this arc. We also have $\sum_{i=1}^{n} \delta_{i1} \leq k$ and $\sum_{i=1}^{n} \delta_{i2} \leq k$ and therefore $\sum_{i=1}^{n} t_i = s_1 \sum_{i=1}^{n} \delta_{i1} + s_2 \sum_{i=1}^{n} \delta_{i2} \leq (s_1+s_2)\, k$. Accordingly, the total flow in (R, Si) does not exceed the capacity of this arc. Since σ is feasible the total flow leaving T_i is equal to τ_i.

Let h be a flow pattern with value $\bar{\bar{F}} = \sum_{i=1}^{n} \tau_i$. Our aim is to show that within each region R we are able to schedule the tasks which, according to h, are to obtain service in R. Let t_i denote the service (not the actual service time) to be obtained by task T_i in region R, i.e., the flow in (T_i, R). We have $t_i \le s_1 k$ for $i=1,\dots,n$ (k is the length of R) and therefore (a) $\max_i t_i \le s_1 k$. Moreover, (b) $\sum t_i \le s_1 k + s_2 k$. Conditions (a) and (b) are known to be necessary and sufficient for the existence of a preemptive schedule of length k on two machines with speeds $s_1 \ge s_2$ and tasks with service times $t_1,\dots,t_n$ [GS,HLS]. □

The worst-case time complexity of an algorithm based on Lemma 2 is $O(n^3)$ since there is a linear-time algorithm for the construction of the schedule in each region R [GS].

References

[Mc] McNaughton, R. "Scheduling with Deadlines and Loss Functions," Management Science, 12, 7 (1959) pp. 1-12.

[EK] Edmonds, J. and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," JACM, V19, N2, April 1972, pp. 248-264.

[GS] Gonzalez, T. and Sahni, S. "Preemptive Scheduling of Uniform Processor Systems," TR 76-5, Computer Science Dept., University of Minnesota, May 1976.

[HLS] Horvath, E. C., Lam, S. and Sethi, R. "A Level Algorithm for Preemptive Scheduling," Fifth Symposium on Operating System Principles, Austin, Texas, Nov. 1975.