V. J. Rayward-Smith
D. Rebane

UET flow shop scheduling with delays


<http://www.numdam.org/item?id=ITA_1996__30_1_23_0>
Abstract. - F|UET, delays |C_{max} is introduced and shown to be NP-complete.

Résumé. - Le problème du flow shop avec des temps de transport est introduit. Il est montré qu'il est NP-difficile même si les temps opératoires sont unitaires.

1. INTRODUCTION

The usual flow shop problem can be described as follows.

Given are a set of n jobs and a set of m machines. Each machine can handle at most one job at a time and each job can be processed by at most one machine at a time. Each job consists of m tasks indexed by 1,..., m and the i-th task of a job precedes its (i + 1)-th task for i = 1,..., m − 1. Further, the i-th task of the j-th job has to be carried out on the i-th machine, during an uninterrupted period of a given length of time, l_{ij}. The purpose is to find a schedule of all the jobs which minimises the overall completion time.

Flow shop scheduling is shown to be NP-complete in the strong sense [1], even for the case m = 3. However, for the special case m = 2, there exists a polynomial time algorithm [2].

In this paper, we introduce the concept of an interprocessor time delay. This models the situation where there is a time delay when a job is transferred from one machine to another.
Let $d_{ij}, 1 \leq i \leq m - 1, 1 \leq j \leq n,$ denote the time delay encountered in transferring job $j$ from machine $i$ to machine $i + 1$ and $c_{ij}$ and $s_{i+1,j},$ respectively, denote the completion time of job $j$ on machine $i$ and the starting time of job $j$ on machine $i + 1.$ Thus, the length of the $i$-th task of job $j$ which must be scheduled on machine $i$ is given by

$$l_{ij} = c_{ij} - s_{ij} + 1.$$ 

If in a given schedule,

$$s_{i+1,j} \geq c_{ij} + d_{ij}, \quad \text{for} \quad 1 \leq i \leq m - 1, 1 \leq j \leq n,$$

then the schedule is called valid.

The delays are uniform if $d_{ij} = d$ for $1 \leq i \leq m - 1$ and $1 \leq j \leq n; \quad$ otherwise the delays are non-uniform and, in general, this is the case we will be considering.

By setting $d_{ij} = 0,$ for all $1 \leq i \leq m - 1, 1 \leq j \leq n,$ it follows immediately from the NP-completeness of flow shop that flow shop with delays is NP-complete in the strong sense for any fixed $m \geq 3.$ In [5] it is shown that the problem of flow shop with delays is NP-complete in the strong sense even for $m = 2.$ However, if this problem is for a permutation flow shop, i.e. the order of the jobs is the same on all machines, then it can be solved in polynomial time [4].

When all the processing times are unit execution times (UET), the optimal flow shop schedule might not be achieved by a permutation flow shop nor by greedily ordering the jobs by nonincreasing delays even for $m = 2.$ For example, consider four UET jobs on two machines with delays 5, 3, 3 and 1. The optimal schedule is of length 8 but the optimal permutation schedule is of length 10 and the greedy schedule is of length 9.

In section 2 of this paper, we prove that the problem of scheduling UET jobs with arbitrary delays in a (non-permutation) flow shop becomes NP-complete if we allow an arbitrary number of processors. The complexity of UET flow shop scheduling with delays for fixed $m \geq 2$ is open.

We summarise that the two machine case is in $P$ but we have been unable to prove this. A useful observation is that, for the two machine case only, there exists a valid schedule of optimal length such that the $n$ jobs are processed continuously on machine 1 and continuously on machine 2. Moreover, for
the two machine case, it is relatively straightforward to establish the bound

\[ \omega_{opt} \geq \max \left\{ \left[ \sum_{j=1}^{k} \frac{d_j}{k} \right] + k + 1 : 1 \leq k \leq n \right\}. \]

However, this bound is not tight [3]; consider six jobs with delays 4, 4, 4, 0, 0, 0. The optimal schedule has length 10 which is greater than the bound of 9 given by the formula.

2. THE NP-COMPLETENESS RESULT

In this section, we prove the following decision problem is NP-complete.

**UET FLOW SHOP WITH DELAYS (FUD)**

**Instance:** number \( p \in \mathbb{Z}^+ \) of processors, set \( J \) of jobs, for each job \( j \in J \) and \( 1 \leq k < p \), a delay \( d(j, k) \in \mathbb{Z}^+ \), and an overall deadline \( D \in \mathbb{Z}^+ \).

**Question:** Is there a valid flow shop schedule of the jobs in \( J \) meeting the deadline where each job \( j \in J \) has an associated UET task on each processor and such that for each \( j \in J \) and each \( 1 \leq k < p \), if \( j \) is processed at time \( t \) on processor \( k \) then it is processed at time \( \geq t + d(j, k) + 1 \) on processor \( k + 1 \)?

We will show that FUD is NP-complete. Our first step is to show that the following well-known NP-complete problem can be polynomially transformed into FUD.

**VERTEX COVER (VC)**

**Instance:** Graph \( G = (V, E) \) and a positive integer, \( k < |V| \).

**Question:** Is there a vertex cover of size \( \leq k \) for \( G \), i.e., a subset \( V' \subset V \) with \( |V'| \leq k \) where each edge in \( E \) is adjacent to some element in \( V' \)?

**Lemma 1:** \( VC \preceq FUD \).

**Proof:** Let an instance of VC comprise a graph, \( G = (V, E) \) with \( |V| = n \), \( |E| = m < n^2 \) and a positive integer \( k < n \). Assume \( V = \{v_1, v_2, \ldots, v_n\} \) and \( E = \{e_1, e_2, \ldots, e_m\} \).
We construct an instance of FUD as follows.

\[ p = 2m + 3, \]
\[ J = \{ x, b_1, b_2, ..., b_k, c_1, c_2, ..., c_{k+1} \} \cup V, \]
\[ d(x, 1) = k + 1, \]
\[ d(c_i, 1) = 3k + 1, 1 \leq i \leq k + 1, \]
\[ d(b_i, 1) = 0, 1 \leq i \leq k, \]
\[ d(v_i, 1) = 0, 1 \leq i \leq n, \]

and then, for \( 1 \leq r \leq m, \)
\[ d(x, 2r) = 2k + 1, \]
\[ d(c_i, 2r) = k, 1 \leq i \leq k + 1, \]
\[ d(b_i, 2r) = k - 1, 1 \leq i \leq k, \]
\[ d(x, 2r + 1) = 0, \]
\[ d(c_i, 2r + 1) = k + 1, 1 \leq i \leq k + 1, \]
\[ d(b_i, 2 + 1) = k + 2, 1 \leq i \leq k, \]

and, for \( 1 \leq i \leq n, \) we define
\[ d(v_i, 2r) = 0, \text{ if } v_i \text{ is adjacent to } e_r, \]
\[ k, \text{ otherwise} \]

and
\[ d(v_i, 2r + 1) = 2k, \text{ if } v_i \text{ is adjacent to } e_r, \]
\[ k, \text{ otherwise}. \]

Finally,
\[ d(x, 2m + 2) = 4k + n + 3, \]
\[ d(c_i, 2m + 2) = 2k + n + 3 - 2i, 1 \leq i \leq k + 1, \]
\[ d(b_i, 2m + 2) = 3k + n + 2 - 2i, 1 \leq i \leq k, \]
\[ d(v_i, 2m + 2) = 0, 1 \leq i \leq n. \]

The deadline, \( D, \) is set by
\[ D = 5k + 2mk + 3m + n + 7. \]

Since \( k \) is \( O(n) \), this is clearly a polynomial transformation.
Our first observation concerns \( x \). The total delays for \( x \) are

\[
(k + 1) + m (2k + 1) + 4k + n + 3.
\]

The total computation time for job \( x \), as for all jobs, is \( p = 2m + 3 \). The sum of these two values is exactly \( D \). Thus, the deadline is met iff \( x \) is the first job processed on machine 1 and the last job processed on machine \( p \). Moreover, the processing time on every intervening machine is also determined uniquely by the delays.

Now, we consider the job \( c_i \). This has a total delay of

\[
3k + 1 + m (2k + 1) + 2k + n + 3 - 2i
= 5k + 2mk + m + n + 4 - 2i
\]

and a processing time of \( p = 2m + 3 \). The sum of these is

\[
5k + 2mk + 3m + n + 7 - 2i = D - 2i.
\]

A simple induction argument can then be used to determine that the deadline is met iff \( c_i \) is processed at time \( 1 + i \) on machine 1 and at time \( D - i \) on machine \( p \). Moreover, the processing times of each \( c_i \) on every intervening machine are also determined uniquely by the delays.

Next, consider the job, \( b_i \). The earliest it can be processed on machine 1 is \( k + 3 \) and the latest it can be processed on machine \( p \) is

\[
D - k - 2 = 4k + 2mk + 3m + n + 5.
\]

The total of the delays for \( b_i \) is

\[
m (2k + 1) + 3k + n + 2 - 2i.
\]

Adding the processing time of \( 2m + 3 \) gives a total of

\[
3k + 2mk + 3m + n + 5 - 2i.
\]

Again, a simple induction argument shows that \( b_i \) must be processed at time \( k + 2 + i \) on machine 1 and at time \( D - k - i - 1 \) on machine \( p \) with all the intervening times uniquely determined by the delays.
The necessary scheduling of the job $x$ and those of types $b$ and $c$ is described in figure 1. This shows that a valid schedule of all these jobs is achievable within the deadline.

We note that on machine $2r$, $(1 \leq r \leq m + 1)$, $x$ is processed at time

$$t_{2r} = 2r + \sum_{j=1}^{2r-1} d(x, j)$$

$$= 2r + k + 1 + (r - 1)(2k + 1)$$

$$= 3r + 2rk - k,$$

$b_i$ is processed at time

$$k + 1 + i + 2r + \sum_{j=1}^{2r-1} d(b_i, j)$$

$$= k + 1 + i + 2r + (r - 1)(2k + 1)$$

$$= t_{2r} + i$$

and $c_i$ is processed at time

$$i + 2r + \sum_{j=1}^{2r-1} d(c_i, j)$$

$$= i + 2r + 3k + 1 + (r - 1)(2k + 1)$$

$$= t_{2r} + 2k + i.$$

On machine $2r + 1$,

$b_i$ is processed at $t_{2r} + k + i$,

$x$ is processed at $t_{2r} + 2k + 2$, and

$c_i$ is processed at $t_{2r} + 3k + i + 1$.

On machine $2r + 1$, spare processing time is thus available in a one unit slot at time $t_{2r} + 2k + 1$ and in a $(k - 1)$ continuous block, $t_{2r} + 2k + 3, \ldots, t_{2r} + 3k + 1$. On machine $2m + 3$, all the available times $\geq D - 2k - 1$ are allocated jobs. The lastest a job in $V$ could be processed on machine $2m + 3$ is

$$D - 2k - 2 = 3k + 2mk + 3m + n + 5$$
and hence on machine $2m + 2$ is

$$3k + 2mk + 3m + n + 4 = 3(m + 1) + 2(m + 1)k - k + 2k + 2 + n - 1 = t_{2m+2} + |J| - 1.$$ 

Thus on machine $2m + 2$, the block of size $k$ between the $b$-jobs and the $c$-jobs must be allocated to $k$ jobs in $V$. The remaining $n - k$ jobs in $V$ must be processed immediately after $c_{k+1}$. This partitions the jobs in $V$ into two sets $V_1$ and $V_2$. The $k$ jobs in $V_1$ are processed on machine $2m + 2$ in the early block of size $k$.

If an element $v \in V$ is processed at time $< t_{2r}$ on machine $2r$ for any $1 < r \leq m$ then, since the $b$-block on machine $2r - 1$ is fixed, this $v$ must be processed before that block on machine $2r - 1$ and hence at time $< t_{2r-2}$ on machine $2r - 2$. Hence, we can deduce that such a $v$ would be processed
at time $< t_2$ on machine 2 which is impossible. We thus know every $v \in V$
is processed at time $> t_{2r}$ on machine $2r$ for all $1 \leq r \leq m + 1$.

Having established that the block of size $k$ on machine $2m + 2$ must
be allocated to $V_1$, an induction argument can then be used to show that
$V_1$ always occupies the $k$ locations between the $b$-block and the $c$-block on
machines $2m$, $2m - 2$, ..., $2$.

Now, consider machine $2r + 1$ $(1 \leq r \leq m)$. The single, isolated location
between the location allocated to $b_k$ and that allocated to $x$ can be used iff
$V_1$ contains a vertex adjacent to $e_r$. A schedule is valid iff this location is
used. Hence, we have a valid schedule within the deadline iff $V_1$ is a vertex
cover. Thus, the lemma is established.

It is now easy to establish

**THEOREM 1:** FUD is NP-complete.

**Proof:** Having established that an NP-complete problem polynomially
transforms to FUD, all we need to establish is that FUD $\in$ NP.

Given the $|J|$ jobs and $p$ machines, we simply guess an integer,
$1 \leq t_j, r \leq D + r - p$, for each $j \in J$, $1 \leq t \leq p$. Then, in polynomial
time, we check that
1. $t_j, r = t_j', r \Rightarrow j = j'$, for all $1 \leq r \leq p$, and
2. $t_j, r + 1 \geq t_j, r + d(j, r)$ for each $j \in J$ and $1 \leq r < p$.

An instance of FUD is a yes-instance iff there is some guess which passes
all these checks.

**REFERENCES**

2. S. JOHNSON, Optimal two and three stage production schedules with set-up times
4. P. L. MAGGU and G. Das, On $2 \times n$ sequencing problem with transportation times
5. R. J. M. VAESSENS and M. DELL'AMICO, Flow and open shop on two machines
   with transportation times and machine independent processing times is NP-hard,
   *unpublished manuscript*, 1995.