Preemptive Multi-Machine Scheduling of Equal-Length Jobs to Minimize the Average Flow Time

Philippe Baptiste\textsuperscript{1}  Marek Chrobak\textsuperscript{2}  Christoph Dürr\textsuperscript{3}  Francis Sourd\textsuperscript{4}

\textsuperscript{1}CNRS and LIX, Ecole Polytechnique, Palaiseau
\textsuperscript{2}CS dep, University of California, Riverside
\textsuperscript{3}LRI, University of Paris-11
\textsuperscript{4}CNRS and LIP6, University of Paris-6

MAPSP, 2005
The Problem $P|r_j;\text{pmtn; } p_j = p| \sum C_j$

- **input**: $n, p, r_1, \ldots, r_n, m$
- **means**: $n$ jobs with equal processing time $p$
- **job j cannot be scheduled before its release time** $r_j$
- **$m$ parallel identical machines**
- **output**: a preemptive schedule with minimizes average completion time
### related problems

- For $m = 2$ solvable in time $O(n \log n)$  
  [Herrbach, Leung, 1990]

- For arbitrary processing times $p_j$ it is binary NP-hard  
  [Du, Leung, Young, 1990]

- ... it is even unary NP-hard [Brucker, Kravchenko, 2004]

### this problem

- [Brucker, Kravchenko, 2004] showed it can be solved with:
  - Sort jobs $r_1 \leq \ldots \leq r_n$ in $O(n \log n)$
  - Solve a linear program of size $O(n^3)$
  - Do some preprocessing in $O(n^3)$

- We show it can be solved directly with a linear program of size $O(nm)$
For $m = 2$, the problem is solvable in time $O(n \log n)$ [Herrbach, Leung, 1990]. For arbitrary processing times $p_j$, it is binary NP-hard [Du, Leung, Young, 1990]. Even unary NP-hard [Brucker, Kravchenko, 2004].

For this problem, [Brucker, Kravchenko, 2004] showed it can be solved with:

- Sort jobs $r_1 \leq \ldots \leq r_n$ in $O(n \log n)$
- Solve a linear program of size $O(n^3)$
- Do some preprocessing in $O(n^3)$

We show it can be solved directly with a linear program of size $O(nm)$.
related problems

- for \( m = 2 \) solvable in time \( O(n \log n) \) [Herrbach, Leung, 1990]
- for arbitrary processing times \( p_j \) it is binary NP-hard [Du, Leung, Young, 1990]
- \( \ldots \) it is even unary NP-hard [Brucker, Kravchenko, 2004]

this problem

- [Brucker, Kravchenko, 2004] showed it can be solved with
  - sort jobs \( r_1 \leq \ldots \leq r_n \) in \( O(n \log n) \)
  - solve a linear program of size \( O(n^3) \)
  - do some preprocessing in \( O(n^3) \)
- we show it can be solved directly with a linear program of size \( O(nm) \)
related problems

- for $m = 2$ solvable in time $O(n \log n)$ [Herrbach, Leung, 1990]
- for arbitrary processing times $p_j$ it is binary NP-hard [Du, Leung, Young, 1990]
- ... it is even unary NP-hard [Brucker, Kravchenko, 2004]

this problem

- [Brucker, Kravchenko, 2004] showed it can be solved with
  - sort jobs $r_1 \leq \ldots \leq r_n$ in $O(n \log n)$
  - solve a linear program of size $O(n^3)$
  - do some preprocessing in $O(n^3)$
- we show it can be solved directly with a linear program of size $O(nm)$
related problems

- for $m = 2$ solvable in time $O(n \log n)$ [Herrbach, Leung, 1990]
- for arbitrary processing times $p_j$ it is binary NP-hard [Du, Leung, Young, 1990]
- ... it is even unary NP-hard [Brucker, Kravchenko, 2004]

this problem

- [Brucker, Kravchenko, 2004] showed it can be solved with
  - sort jobs $r_1 \leq \ldots \leq r_n$ in $O(n \log n)$
  - solve a linear program of size $O(n^3)$
  - do some preprocessing in $O(n^3)$
- we show it can be solved directly with a linear program of size $O(nm)$
Definition of a normal schedule

- every job is scheduled in at most one interval on every machine
- and the intervals are ordered by machines
- the executions on a fixed machine are ordered by jobs (suppose \( r_1 \leq \ldots \leq r_n \))

Our main Theorem

Every schedule can be put in normal form without increasing \( \sum C_j \)
The resulting linear program

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=1}^{n} C_{j,1} \\
\text{subject to} & \quad -S_{j,m} \leq -r_j \quad j = 1, \ldots, n \\
& \quad \sum_{q} (C_{j,q} - S_{j,q}) = p \quad j = 1, \ldots, n \\
& \quad S_{j,q} - C_{j,q} \leq 0 \quad j = 1, \ldots, n, \quad q = 1, \ldots, m \\
& \quad C_{j,q} - S_{j,q-1} \leq 0 \quad j = 1, \ldots, n, \quad q = 2, \ldots, m \\
& \quad C_{j,q} - S_{j+1,q} \leq 0 \quad j = 1, \ldots, n - 1, \quad q = 1, \ldots, m
\end{align*}
\]
Let $I$ be the time set where exactly one of the jobs $i, j$ ($r_i \leq r_j$) is scheduled.

The reduction of $i, j$ consists of scheduling only $i$ in the first half of $I$ and only $j$ in the second half.

$C_i + C_j$ does not increase.
Let $I$ be the time set where exactly one of the jobs $i, j$ ($r_i \leq r_j$) is scheduled.

The reduction of $i, j$ consists of scheduling only $i$ in the first half of $I$ and only $j$ in the second half.

$C_i + C_j$ does not increase.
Simplifying Assumption

- All start-, preemption- and completion-times are integer.
- In every slot \([t, t + 1)\) 1st job is assigned to 1st machine, 2nd job to 2nd machine...
Proof

**Lemma** After a finite number of reductions any schedule is in normal form.

**Proof**

- The discrete vector \((H(1), \ldots, H(n))\) decreases lexicographically with each reduction, where \(H(i) = \text{sum of integer times } t \text{ where } i \text{ is scheduled}\).
- If the number of jobs \(\leq j\) scheduled in \([t, t + 1)\) for \(t \leq r_j\) increases, then a reduction is possible.
## More related problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P2_{</td>
<td>r_j;\ \text{pmtn}; \ p_j = p</td>
</tr>
<tr>
<td>$P_{</td>
<td>r_j; \ \text{pmtn}; \ p_j = p</td>
</tr>
<tr>
<td>$P_{</td>
<td>r_j; \ \text{pmtn}</td>
</tr>
<tr>
<td>$P_{</td>
<td>\ \text{pmtn}; \ p_j = p</td>
</tr>
<tr>
<td>$P_{</td>
<td>r_j; \ p_j = p</td>
</tr>
<tr>
<td>$1_{</td>
<td>r_j; \ \text{pmtn}; \ p_j = p</td>
</tr>
<tr>
<td>$P_{</td>
<td>r_j; \ \text{pmtn}; \ p_j = p</td>
</tr>
</tbody>
</table>