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

Philippe Baptiste[1] Marek Chrobak[2] Christoph Dürr[3] Francis Sourd[4]

[1]CNRS and LIX, Ecole Polytechnique, Palaiseau

[2]CS dep, University of California, Riverside

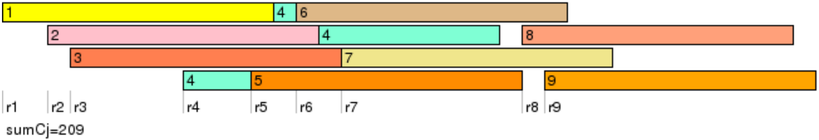
[3]LRI, University of Paris-11

[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



History

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 $O(nm)$

History

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 $O(nm)$

History

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 $O(nm)$

History

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 $O(nm)$

History

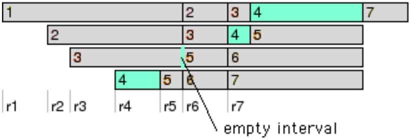
related problems

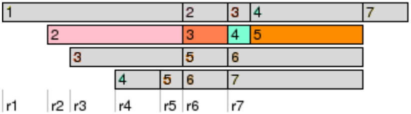
- for $m = 2$ solvable in time $O(n \log n)$ [Herrbach,Leung,1990]
- for arbitrary processing times p_j it is binary NP-hard [Du,Leung,Young,1990]
- ...it is even unary NP-hard [Brucker,Kravchenko,2004]

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



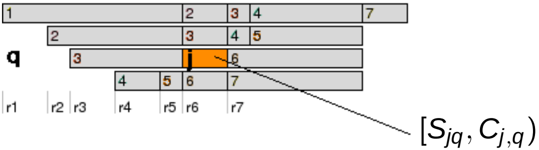


- every job is scheduled in at most one interval on every machine
- and the intervals are ordered by machines
- the executions on a fixed machine are ordered by jobs (suppose $r_1 \leq \ldots \leq r_n$)

Our main Theorem

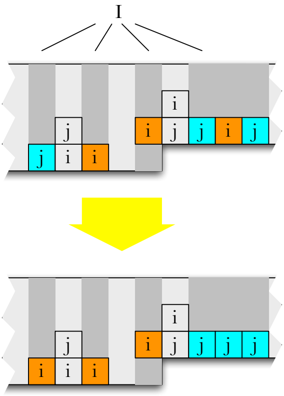
Every schedule can be put in normal form without increasing $\sum C_j$

The resulting linear program



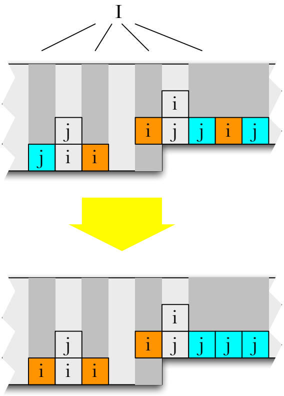
$$
\begin{array}{lrll}
\text{minimize} & \sum_{j=1}^{n} C_{j,1} & & \\
\text{subject to} & -S_{j,m} & \leq -r_j & j = 1, \ldots, n \\
& \sum_q (C_{j,q} - S_{j,q}) & = p & j = 1, \ldots, n \\
& S_{j,q} - C_{j,q} & \leq 0 & j = 1, \ldots, n,\ q = 1, \ldots, m \\
& C_{j,q} - S_{j,q-1} & \leq 0 & j = 1, \ldots, n,\ q = 2, \ldots, m \\
& C_{j,q} - S_{j+1,q} & \leq 0 & j = 1, \ldots, n-1,\ q = 1, \ldots, m
\end{array}
$$

Reduction



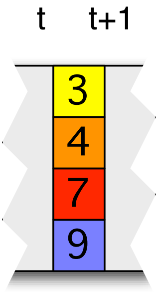
- Let I be the time set where exactly one of the jobs i, j ($r_i \leq r_j$) is scheduled.
- The reduction of i, j consists of scheduling only i in the first half of I and only j in the second half.
- $C_i + C_j$ does not increase.

Reduction



- 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



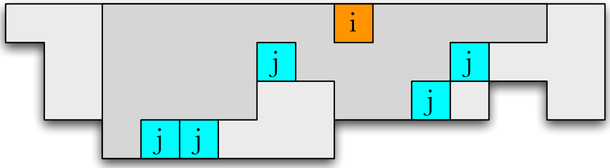
- all start-, preemption- and completion-times are integer.
- in every slot $[t, t+1)$ 1st job is assigned to 1st machine, 2nd job to 2nd machine...

Proof

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

Proof

- The discrete vector $(H(1), \dots, H(n))$ decreases lexicographically with each reduction,
  where $H(i) =$ sum of integer times t where i 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

Problem	Complexity
$P2 \mid r_j; \text{pmtn}; p_j = p \mid \sum C_j$	$O(n \log n)$ [Herrbach,Leung,1990]
$P \mid r_j; \text{pmtn}; p_j = p \mid \sum C_j$	this talk
$P \mid r_j; \text{pmtn} \mid \sum C_j$	binary NP-complete [Du,Leung,Young,1990]
$P \mid \text{pmtn}; p_j = p \mid \sum C_j$	solvable by the greedy algorithm (trivial)
$P \mid r_j; p_j = p \mid \sum C_j$	solvable by the greedy algorithm (trivial)
$1 \mid r_j; \text{pmtn}; p_j = p \mid \sum w_j C_j$	open
$P \mid r_j; \text{pmtn}; p_j = p \mid \sum w_j C_j$	unary NP-complete [Leung,Young,1990]