Just-in-Time preemptive scheduling around a common due date

Francis Sourd

LIP6
CNRS - Université Paris 6

GOThA
Paris - 15 avril 2005
Preemption and JIT scheduling

- $n$ operations (processing time $p_i$).
- **Preemption** is allowed.
- Find a **one-machine schedule** that minimize the total cost.
- How to define **job costs** to model the **Just-in-Time** philosophy?
Early-tardy completion

- Earliness-tardiness penalties $\alpha_i E_i + \beta_i T_i$
Early-tardy completion

- Earliness-tardiness penalties $\alpha_i E_i + \beta_i T_i$

- But naive preemption...

- ...results in earliness relaxation
Position costs

- Cost functions $f_i : [0, \infty) \rightarrow \mathbb{R}$
- Cost $f_i(t)dt$ when $i$ is processed between $t$ and $t + dt$.

Cost of job $i$ is

$$\int_0^\infty x_i(t)f_i(t)dt$$

$x_i(t)$ indicator function of the processing of job $i$.

$$\int_0^\infty x_i(t)dt = p_i$$
Objective function

- Minimize

\[ \sum_{i=1}^{n} \int_{0}^{\infty} x_i(t)f_i(t) dt \]

- Problem notation

1|pmtn| \( \sum f \)
Preemption at integer time points

[Sourd and Kedad-Sidhoum, JoS 2003]

- interruption only at integer time points
- tasks divided into unit execution time operations
- costs $c_{it} = \int_{t}^{t+1} f_i(t) dt$ for scheduling a UET operation of job $i$ in $[t, t+1)$

but the size of the relaxed problem is pseudopolynomial
Dual problem

\[ \min \sum_{i=1}^{n} \int_{0}^{\infty} x_i(t) f_i(t) dt \]

s.t. \[ \int_{0}^{\infty} x_i(t) dt = p_i \]
\[ \sum_i x_i(t) \leq 1 \]
Dual problem

\[
\min \sum_{i=1}^{n} \int_{0}^{\infty} x_i(t) f_i(t) dt \\
\text{s.t.} \quad \int_{0}^{\infty} x_i(t) dt = p_i \times u_i \\
\quad \sum_{i} x_i(t) \leq 1
\]
Dual problem

\[ L(u) = \min \sum_{i=1}^{n} \int_{0}^{\infty} x_i(t)(f_i(t) - u_i)dt + \sum_i u_i p_i \]

s.t.

\[ \sum_i x_i(t) \leq 1 \]
Dual problem

\[ L(u) = \min \sum_{i=1}^{n} \int_{0}^{\infty} x_i(t)(f_i(t) - u_i)dt + \sum u_i p_i \]
\[ \text{s.t.} \]
\[ \sum x_i(t) \leq 1 \]
Dual problem

[Sourd, INFORMS JoC, 2004]

\[ \pi_2(u) = \pi_1(u) + \pi_3(u) \]

- **No duality gap**
- **At the optimum,**
  \[ (\pi_1(u), \pi_2(u), \ldots, \pi_n(u)) = (p_1, p_2, \ldots, p_n) \]
- **Polynomial** with the ellipsoid method
Motivation

- Study special easier cases
- Better understanding of this new criterion
- Efficient strongly polynomial algorithms
Today’s problem

- Common due date $d$ for each job
- Cost function

$$f_i(t) = \alpha_i \max(0, d - t) + \beta_i \max(0, t - d)$$
No earliness — $d = 0$

- $f_i(t) = \beta_i t$
- Larger slope first
No earliness — $d = 0$

- $f_i(t) = \beta_i t$
- Larger slope first
No earliness — \( d = 0 \)

- \( f_i(t) = \beta_i t \)
- Larger slope first
No earliness — $d = 0$

- $f_i(t) = \beta_i t$
- Larger slope first
No earliness — $d = 0$

- $f_i(t) = \beta_i t$
- Larger slope first
Basic properties of the solution

- An optimal schedule
  - starts at $t \leq d$
  - ends at $t + P \geq d$ with $P = \sum_i p_i$
  - no idle time in between the tasks
- the tardy parts of jobs are sorted according to the $\beta_i$
- the early parts of jobs are sorted according to the $\alpha_i$
Rationale of the algorithm

- Let $f(t)$ be the optimal cost for scheduling all the jobs in $[t, t + P)$
- $f$ is convex.
- Minimize the function $f$ when $t$ varies.
- Start with $t = d$ (jobs are all late).
- Compute $f(t - \epsilon)$ from $f(t)$ by maintaining the primal and dual solutions.
- End when the minimum of $f$ is reached.
From $f(t)$ to $f(t - \epsilon)$

**Lemma**

*Only one job ($i^*$) is transferred when $t$ decreases.*

**Sketch of the proof.**

- The jobs in $\bar{E}$ are **completely** early.
- The jobs in $\bar{T}$ are **completely** tardy.
- The dual variables of the job in between $i^*$ do not change.
Selecting the transfered job

- The proof of the previous lemma shows how to select the transfered job according to the dual problem.
Selecting the transfered job

- The proof of the previous lemma shows how to select the transfered job according to the dual problem.
- A primal approach computationally more efficient
Selecting the transferred job

- The proof of the previous lemma shows how to select the transferred job according to the dual problem.
- A primal approach computationally more efficient
- **Marginal transfer cost**
  - if job $i$ is transferred
    \[
    f(t - \epsilon) = f(t) + m_i \epsilon + o(\epsilon)
    \]
  - $m_i = \sum_j \min(\alpha_j, \alpha_i) p_j^– - \min(\beta_j, \beta_i) p_j^+$
- Select the job with the **smallest** marginal transfer cost.
- The variation of $m_i$ is (piecewise) linear.
  \[
  m_i(t - \epsilon) = m_i(t) + (\min(\alpha_i, \alpha_i^*) + \min(\beta_i, \beta_i^*)) \epsilon
  \]
Events

- **Discretize** the “continuous” procedure
- **Classes of events**
  1. Transfer if job $i^*$ completed
  2. Another job becomes critical
  3. $t = 0$
  4. Minimum of $f$ is reached
- As the variation of the marginal costs are linear, the distance between the current event and the next event can be easily computed.
Number of events

Lemma

The transfer of a job can only be interrupted by a wholly late job.

Corollary

There are $O(n)$ events.
Complexity

**Theorem**

*The algorithm runs in $O(n^2)$ time.*

**Proof.**

- There are $O(n)$ events
- Marginal transfer costs are updated in $O(n)$ time.
- Next event is calculated in $O(n)$ time.
Conclusion

▶ An $O(n^2)$ algorithm for the common due date problem
▶ Release dates, deadlines ?
▶ Non common due dates ?
▶ Lower bound for the non-preemptive problem.