Scheduling with Precedence Constraints

James Jolly and Pratima Kolan

November 25, 2009
Precedence Graphs in Task Scheduling

- typically DAGs
- vertices are tasks
  - processing time $p_i$
  - weight $w_i$
- edges are data dependencies
  - $i \rightarrow j$, $i$ precedes $j$
Precedence Graphs in Task Scheduling
Precedence Graph Scheduling Problems

Objectives:

▶ minimize makespan
▶ minimize weighted completion time
▶ maximize throughput

Consider:

▶ release times
▶ resource constraints
Focus: Minimize Weighted Completion Time

- $G$ is a DAG
- $n$ tasks
- task $i$ has weight $w_i$
- task $i$ finishes at $c_i$

minimize $\sum_{i=1}^{n} w_i c_i$

under precedence constraints

(NP-Hard)
Our Goal

- show two approximations
- construct single-machine schedule
- convert single-machine schedule to multi-machine schedule
Single-Machine Scheduling With Precedence

- breaks tasks into groups (P-time)
- ranks groups
- schedules each group in order of increasing rank (P-time, $\alpha = 2$)
Sub-DAG Rank

- given collection of tasks $T = \{t_1, t_2, ..., t_k\}$
- importance of scheduling $T$ first

\[
R(T) = \frac{\sum_{i=1}^{k} p_i}{k \sum_{i=1}^{k} w_i}
\]
Precedence Closed Sub-DAG

- every task inside only depends on other tasks inside
Minimal Rank Precedence Closed Sub-DAG, G*

- Properties:
  - feasible schedule for $G^*$ is 2OPT
  - there exists an optimal schedule $S$ of $G$ where the optimal schedule for $G^*$ comes as a segment starting at time 0
Approximation schedule for $G^*$

- if $G^*$ has rank $\alpha$, then any subgraph of $G^*$ has rank higher than $\alpha$

\[
\forall j \in G^*, \quad \frac{\sum_{1 \leq i \leq j} p_i}{\sum_{1 \leq i \leq j} w_i} \geq \alpha
\]

\[
\text{OPT} = \sum_j w_j C_j \\
\geq \sum_j w_j \sum_{i \leq j} \alpha w_i = \alpha \sum_j (w_j)^2 + \sum_{i \leq j} w_i w_j \\
= \alpha (W(G^*))^2 - \frac{(W(G^*))^2}{2} = \frac{\alpha (W(G^*))^2}{2} = \frac{P(G^*) W(G^*)}{2}
\]

- any schedule with no idle time has weighted completion time of at most $P(G) W(G)$
Overview of the Algorithm
Approximation Factor

- Total weighted completion time of $G$ is:
  \[ \gamma(G^*) + p(G^*)w(G - G^*) + \gamma(G - G^*) \]

- How do we find $G^*$?
**G* Construction**

construct a graph $G_\lambda$
solve it by finding min-cut of the graph
use min-cut to find sub-DAG of rank at most $\lambda$

- vertex set $V = T \cup \{(s, t)\}$
- add an edge from source $s$ to every job with cost on it equal to $\lambda w_i$
- add an edge from every job to the sink $t$ with cost equal to processing time of the job
- for every precedence constraint between two vertices $t_1, t_2$ in $G$, then we add an edge from $t_2$ to $t_1$ having infinite cost
$G^*$ Construction

if there exists any cut $(A, B)$ in $G_\lambda$ whose value is bounded $\lambda w(G)$ then subgraph $A - \{s\}$ is precedence closed and that the rank of $A - \{s\}$ is at most $\lambda$
\( \lambda \) Values

- how do we increase \( \lambda \)?
- \( \lambda_{\text{min}} = \) minimum rank of any vertex
- \( \lambda_{\text{max}} = \) rank of the graph
- perform binary search
Execution Step 1

Here \( \{A, B, C\} \) form a minimal rank precedence closed subgraph.
Execution Step 2

Here $\{D\}$ forms a minimal rank precedence closed subgraph.
Execution Step 3

\{F, E\} is a minimal rank precedence closed subgraph.
{G} forms minimal rank precedence closed subgraph.
Multi-Machine Scheduling With Precedence

- requires feasible single-machine schedule as input
- uses identical machines $M = \langle m_1, m_2, ..., m_k \rangle$
- weighs parallelism increases against input schedule ordering
Delay List Intuition

- schedule lowest rank ready tasks next
- sometimes beneficial to schedule tasks out of order
  - take advantage of an idle processor
  - may bump back an important process
Delay List Algorithm

\[ t = 0 \]
if a machine \( m \) in \( M \) is idle, then:
  if the first task \( i \) in \( V \) is ready:
    schedule \( i \) on \( m \)
    mark all idle time up to start time of task \( i \)
  otherwise:
    scan through \( V \), pick the first task \( i \) that is ready
    if \( \beta \times P_i \leq \text{sum of all unmarked idle time} \)
      schedule task \( i \) on \( m \)
\[ t = t + 1 \]
Choosing $\beta$

- low $\beta$, avoid processor downtime, more out-of-order scheduling
- high $\beta$, accept more downtime, more faithful scheduling
- input schedule quality and weight variance important factors
Conclusions

- can produce approximate single-machine schedules
- single-machine schedules beget multi-machine schedules
- weight and processing time distributions help tuning both
- immense practical significance