Aalto University
School of Science

Department of Computer Science

Combinatorics of Efficient Computations

Approximation Algorithms

Lecture 10: Scheduling Jobs on Parallel Machines

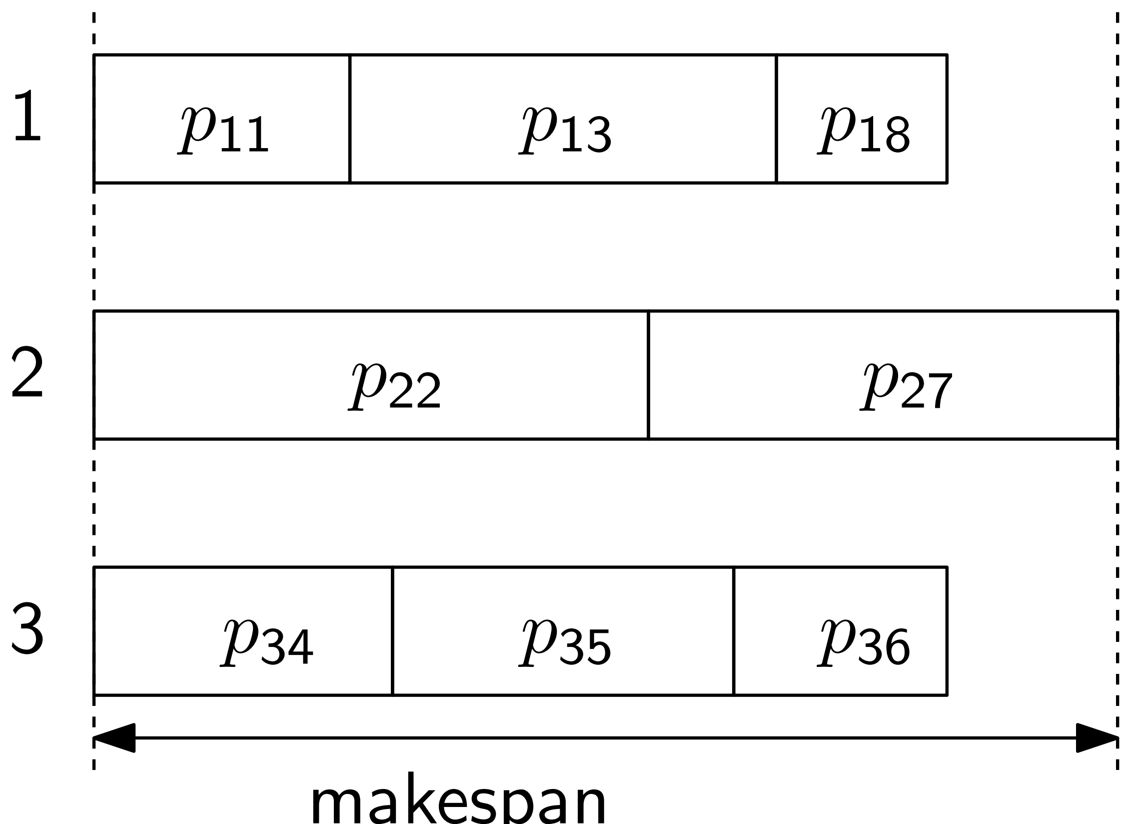
Joachim Spoerhase

2019

Scheduling on Parallel Machines

Given: A set J of **Jobs**, a set M of **machines** and for each $j \in J$ and $i \in M$ the **processing time** $p_{ij} \in \mathbb{N}^+$ of j on i.

Find: A **Schedule** $\sigma \colon J \to M$ of the jobs on the machines, which minimizes the total time to completion (**makespan**), i.e., minimizes the maximum time a machine is in use.



$J = \{1, 2, \ldots, 8\}$

$M = \{1, 2, 3\}$

A natural ILP

$$\begin{aligned} \text{minimize} \quad & t \\ \text{s.t.} \quad & \sum_{i \in M} x_{ij} = 1, & j \in J \\ & \sum_{j \in J} x_{ij} p_{ij} \leq t, & i \in M \\ & x_{ij} \in \{0, 1\}, & i \in M, j \in J \end{aligned}$$

Task: Show that the integrality gap of this ILP is unbounded.

Solution: A job with processing time m and m machines $\rightsquigarrow$ $\mathrm{OPT} = m$ and $\mathrm{OPT}_f = 1$

Parametrized Pruning

Strengthen the ILP $\rightarrow$ implicit (non-linear) constraint:

If $p_{ij} > t$ then set $x_{ij} = 0$

Parameter $T \in \mathbb{N}^+$. Estimate a lower bound on OPT

Define $S_T := \{\, (i,j) \mid i \in M, j \in J, p_{ij} \leq T \,\}$

Define the "pruned" relaxation LP(T)

$$
\begin{array}{lll}
\displaystyle\sum_{i\,:\,(i,j)\in S_T} x_{ij} = 1, & & j \in J \\
\displaystyle\sum_{j\,:\,(i,j)\in S_T} x_{ij} p_{ij} \leq T, & & i \in M \\
x_{ij} \geq 0, & & (i,j) \in S_T
\end{array}
$$

no objective function; just need to determine if a feasible solution exists.

Properties of Extreme-Point Solutions

Use binary search to find the smallest T so that LP(T) has a solution and let T^* be this value of T.

What are the bounds for our search?

Note: $T^* \leq$ OPT

Idea: Round an extreme-point solution of LP(T^*) to a schedule whose makespan is $\leq 2T^*$

LP(T)

$$\sum_{i\colon (i,j)\in S_T} x_{ij} = 1, \qquad j \in J$$

$$\sum_{j\colon (i,j)\in S_T} x_{ij} p_{ij} \leq T, \qquad i \in M$$

$$x_{ij} \geq 0, \qquad (i,j) \in S_T$$

Lem. 1
Each extremepoint solution to LP(T) has at most $m + n$ positive variables where $m = |M|, n = |J|$.

Lem. 2
Any extreme-point solution to LP(T) must set at least $n - m$ jobs integrally.

Extreme-Point Solutions of LP(T)

Def. bipartite graph $G = (J, M, E)$, where
$(j, i) \in E \Leftrightarrow x_{ij} \neq 0$

Let $F \subseteq J$ be the set of fractionally assigned jobs and let
$H := G[F \cup M]$

Note: (i, j) is an edge in $H \Leftrightarrow 0 < x_{ij} < 1$

A matching in H is called F-**perfect**, when it matches every vertex in F.

Key step: Show that H always has an F-perfect matching.

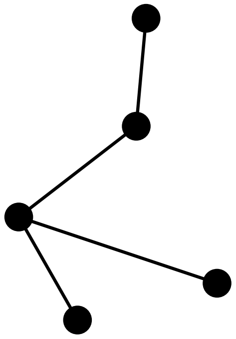
Why is this useful ....?

Algorithm

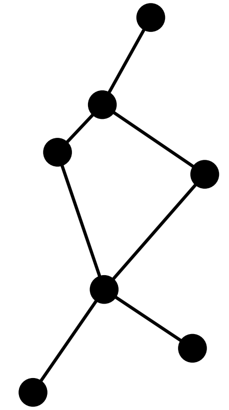
- Assign job j to machine i such that i is the machine minimizing p_{ij}. Let α be the makespan of this schedule.
- By a binary search in the interval $[\frac{\alpha}{m}, \alpha]$, find the smallest value of $T \in \mathbb{Z}^+$ for which LP(T) has a feasible solution and let this value be T^*.
- Find an extreme point solution, say $\mathbf{x}$, to LP(T^*).
- Assign all integrally set jobs to machines as in $\mathbf{x}$.
- Construct the graph H and find a perfect matching P in it (see Lemma 4 later).
- Assign the fractional jobs to machines using P.

Thm. This algorithm is a 2-approximation.
(assuming we have the F-perfect matching)

Pseudo-Trees and -Forests



A connected graph with vertex set V is called a **Pseudo-Tree**, when it has at most $|V|$ edges.



A pseudo-tree is a tree or a tree plus a single edge.

A collection of disjoint pseudo-trees is called a **pseudo-forest**.

Lem. 3 The bipartite graph $G = (J, M, E)$ is a pseudo-forest.
Recall: (by Lem. 1) each extreme point solution has at most $n + m$ non-zero variables.

Lem. 4 The graph H has an F-perfect matching.

Scheduling on Parallel Machines

Thm. There is an LP-based 2-approximation algorithm for the problem of scheduling jobs on unrelated parallel machines.

Is this tight? Yes

Instance m:

- $m^2 - m + 1$ jobs to be scheduled on m machines.
- job j_1 has a processing time of m on all machines,
- all other jobs have unit processing time on each machine.

Optimum: one machine with j_1, and all others spread evenly.

Algorithm:

- LP(T) has no feasible solutions for any $T < m$.
- extreme-pt. solution: assign $1/m$ of j_1 and $m - 1$ other jobs to each machine. $\rightsquigarrow 2m - 1$ makespan.

Scheduling on Parallel Machines

Thm. There is an LP-based 2-approximation algorithm for the problem of scheduling jobs on unrelated parallel machines. The approximation factor is tight.