Algorithms for Hierarchical and Semi-Partitioned Parallel Scheduling

Vincenzo Bonifaci [1]
Gianlorenzo D'Angelo [2]
Alberto Marchetti-Spaccamela [3]

[1] IASI - Consiglio Nazionale delle Ricerche, Rome
[2] Gran Sasso Science Institute, L'Aquila
[3] Sapienza University of Rome

IWES 2017
(published at IEEE IPDPS 2017)

Motivation: Scheduling jobs on computing clusters

Given: a set of computational jobs

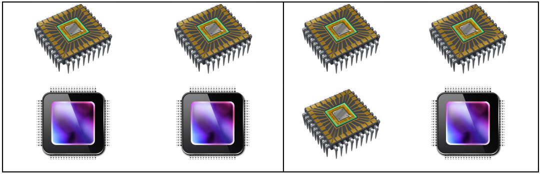
J_1 Multiply.exe J_2 MaxCut.exe J_3 TSP.exe ...

Motivation: Scheduling jobs on computing clusters

Given: a set of computational jobs

J_1	Multiply.exe	J_2	MaxCut.exe	J_3	TSP.exe	...

to be scheduled on clusters of heterogeneous machines (processors)

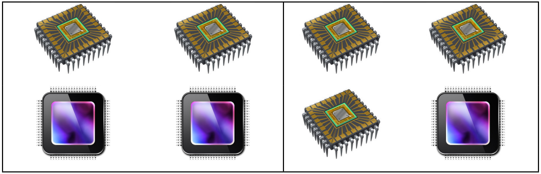


Motivation: Scheduling jobs on computing clusters

Given: a set of computational jobs

J_1 Multiply.exe J_2 MaxCut.exe J_3 TSP.exe ...

to be scheduled on clusters of heterogeneous machines (processors)



Two types of approach: with or without migration

1. Partitioned: Confine each job to a specific machine
2. Global: Allow jobs to migrate between machines (and clusters)

Partitioned scheduling vs. global scheduling

Partitioned scheduling

+ No migration overheads
– More constrained, smaller set of schedulable instances

Global scheduling

+ Less constrained, larger set of schedulable instances
– Migration overheads

Hybrid approaches

- Semi-partitioned scheduling
    - pre-assign some of the jobs to the machines
    - allow global migrations for the rest
- Clustered scheduling:
    - pre-assign each job to a machine cluster
    - allow migrations inside each cluster

Bastoni, Brandenburg & Anderson (2010):
experimental comparison of the trade-offs

Processor affinity

- System interface to restrict the set of processors on which a job may be scheduled
- Widely available across operating systems:
  - Linux: `sched_setaffinity()`
  - FreeBSD: `cpuset_setaffinity()`
  - Windows: `SetThreadAffinityMask()`

Example of affinity mask settings

	mach. 1	mach. 2	mach. 3	mach. 4
job 1	x	x	–	–
job 2	x	x	x	x
job 3	–	–	x	x
job 4	x	–	–	–

Example of affinity mask settings

	mach. 1	mach. 2	mach. 3	mach. 4
job 1	x	x	–	–
job 2	x	x	x	x
job 3	–	–	x	x
job 4	x	–	–	–

- Question: How to set affinity masks to achieve good tradeoffs? And how to model the tradeoffs in the first place?

Our formalization

Idea: allow the processing time to depend on the affinity mask of the job

- jobs $J = \{1, \dots, n\}$
- machines $M = \{1, \dots, m\}$
- a family of admissibile sets $\mathcal{A} \subseteq 2^M$ (the available masks)
- for each $j \in J$, a function $P_j : \mathcal{A} \to \mathbb{Z}_+$
  - monotone: $\alpha \subset \beta \Rightarrow P_j(\alpha) \leq P_j(\beta)$

Our formalization

Idea: allow the processing time to depend on the affinity mask of the job

- jobs $J = \{1, \ldots, n\}$
- machines $M = \{1, \ldots, m\}$
- a family of admissibile sets $\mathcal{A} \subseteq 2^M$ (the available masks)
- for each $j \in J$, a function $P_j : \mathcal{A} \to \mathbb{Z}_+$
  - monotone: $\alpha \subset \beta \Rightarrow P_j(\alpha) \leq P_j(\beta)$

Interpretation: $P_j(\alpha)$ is the processing time of j when j is allowed to migrate over any machine in α

$\Rightarrow$ migration overheads can be embedded in $P_j(\alpha)$, if desired

Our formalization

Idea: allow the processing time to depend on the affinity mask of the job

- jobs $J = \{1, \ldots, n\}$
- machines $M = \{1, \ldots, m\}$
- a family of admissibile sets $\mathcal{A} \subseteq 2^M$ (the available masks)
- for each $j \in J$, a function $P_j : \mathcal{A} \to \mathbb{Z}_+$
  - monotone: $\alpha \subset \beta \Rightarrow P_j(\alpha) \leq P_j(\beta)$

Interpretation: $P_j(\alpha)$ is the processing time of j when j is allowed to migrate over any machine in α
$\Rightarrow$ migration overheads can be embedded in $P_j(\alpha)$, if desired

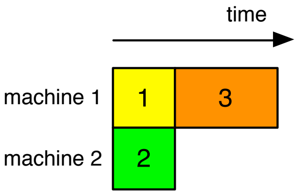
Goal: for each job j, find a set $\bar{\alpha}_j \in \mathcal{A}$ and a schedule of j on $\bar{\alpha}_j$ that minimizes the makespan

An example

$J = \{1, 2, 3\}$
$M = \{1, 2\}$
$\mathcal{A} = \{\{1\}, \{2\}, \{1, 2\}\}$

j	$P_j(\{1\})$	$P_j(\{2\})$	$P_j(\{1,2\})$
1	4	∞	∞
2	∞	4	∞
3	7	7	10

A possible solution:



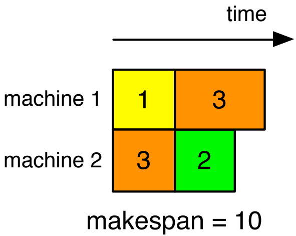
makespan = 11

An example

$J = \{1, 2, 3\}$
$M = \{1, 2\}$
$\mathcal{A} = \{\{1\}, \{2\}, \{1, 2\}\}$

j	$P_j(\{1\})$	$P_j(\{2\})$	$P_j(\{1,2\})$
1	4	∞	∞
2	∞	4	∞
3	7	7	10

A possible solution:



Note: simultaneous parallel processing of the same job is not allowed

Examples with different $\mathcal{A}$

By varying $\mathcal{A}$ we recover classical and newer problems:

Examples with different $\mathcal{A}$

By varying $\mathcal{A}$ we recover classical and newer problems:

- $\mathcal{A} = \{\{1\}, \{2\}, \ldots, \{m\}\}$: unrelated machines without migration

Examples with different $\mathcal{A}$

By varying $\mathcal{A}$ we recover classical and newer problems:

- $\mathcal{A} = \{\{1\}, \{2\}, \ldots, \{m\}\}$: unrelated machines without migration
- $\mathcal{A} = \{M\}$: identical parallel machines with migration

Examples with different $\mathcal{A}$

By varying $\mathcal{A}$ we recover classical and newer problems:

- $\mathcal{A} = \{\{1\}, \{2\}, \ldots, \{m\}\}$: unrelated machines without migration
- $\mathcal{A} = \{M\}$: identical parallel machines with migration
- $\mathcal{A} = \{M, \{1\}, \{2\}, \ldots, \{m\}\}$: semi-partitioned scheduling

Examples with different $\mathcal{A}$

By varying $\mathcal{A}$ we recover classical and newer problems:

- $\mathcal{A} = \{\{1\}, \{2\}, \ldots, \{m\}\}$: unrelated machines without migration
- $\mathcal{A} = \{M\}$: identical parallel machines with migration
- $\mathcal{A} = \{M, \{1\}, \{2\}, \ldots, \{m\}\}$: semi-partitioned scheduling
- $\mathcal{A} = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}\}$: clustered scheduling

Examples with different $\mathcal{A}$

By varying $\mathcal{A}$ we recover classical and newer problems:

- $\mathcal{A} = \{\{1\}, \{2\}, \ldots, \{m\}\}$: unrelated machines without migration
- $\mathcal{A} = \{M\}$: identical parallel machines with migration
- $\mathcal{A} = \{M, \{1\}, \{2\}, \ldots, \{m\}\}$: semi-partitioned scheduling
- $\mathcal{A} = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}\}$: clustered scheduling

In many cases, the family $\mathcal{A}$ is hierarchical, or laminar:

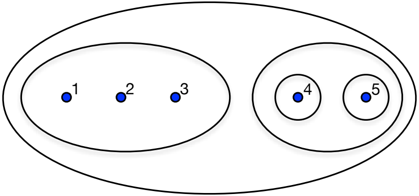
- $\alpha, \beta \in \mathcal{A} \Rightarrow \alpha \subseteq \beta \lor \beta \subseteq \alpha \lor \alpha \cap \beta = \emptyset$

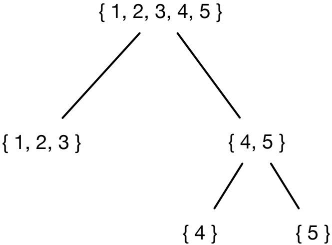
We will assume $\mathcal{A}$ laminar for our results
(the model makes sense even without this assumption)

We call the resulting problem HIERARCHICAL SCHEDULING

Structure of laminar families

Say $\mathcal{A} = \{\{1,2,3\},\{4\},\{5\},\{4,5\},\{1,2,3,4,5\}\}$





There are never "too many" affinity masks: $|\mathcal{A}| \leq 2m$

Hardness of Hierarchical Scheduling

Hierarchical Scheduling generalizes the $R||C_{\max}$ problem (scheduling on unrelated machines)

$\Rightarrow$ Computing solutions of makespan less than $\left(\frac{3}{2} - \epsilon\right) \cdot \mathrm{opt}$ is NP-hard (Lenstra, Shmoys & Tardos 1987)

opt: minimum makespan of a solution
ϵ: arbitrary positive constant

Approximability: our result

Let $\rho \geq 1$. A ρ-approximation algorithm outputs, in polynomial time given any HS instance I, a solution with makespan $\leq \rho \cdot \mathrm{opt}(I)$

Approximability: our result

Let $\rho \geq 1$. A ρ-approximation algorithm outputs, in polynomial time given any HS instance I, a solution with makespan $\leq \rho \cdot \mathrm{opt}(I)$

Main Result (B., D'Angelo, Marchetti-Spaccamela)

Hierarchical Scheduling admits a 2-approximation algorithm.

Related work on unrelated machines

- Lenstra, Shmoys & Tardos (1987): $R||C_{\max}$ admits a 2 approximation

Related work on unrelated machines

- Lenstra, Shmoys & Tardos (1987): $R||C_{\max}$ admits a 2 approximation
  Nothing better than a $2 - \frac{1}{m}$ is known, even today

Outline of the approach

1. Integer Linear Programming formulation of the problem

Outline of the approach

1. Integer Linear Programming formulation of the problem
2. Prove the ILP formulation is exact
   - The ILP constraints are necessary (trivial)
   - Show that given ILP solution $(\mathbf{x}, T)$, one can construct a valid schedule of makespan T using the affinity masks described by $\mathbf{x}$

Outline of the approach

1. Integer Linear Programming formulation of the problem
2. Prove the ILP formulation is exact
    - The ILP constraints are necessary (trivial)
    - Show that given ILP solution $(\mathbf{x}, T)$, one can construct a valid schedule of makespan T using the affinity masks described by $\mathbf{x}$
3. Show how to approximately round the ILP
    - Leverage the LP structure to redistribute the fractional values on the leaves of $\mathcal{A}$
    - Invoke Lenstra-Shmoys-Tardos rounding to get solution $(\bar{\mathbf{x}}, 2T)$ with $\bar{\mathbf{x}}$ integral

ILP formulation for Semi-Partitioned Scheduling

p_{ij}: shorthand for $P_j(\{i\})$
p_{0j}: shorthand for $P_j(M)$
x_{ij}: 1 if j assigned to machine i, 0 otherwise
x_{0j}: 1 if j assigned globally, 0 otherwise

ILP formulation for Semi-Partitioned Scheduling

p_{ij}: shorthand for $P_j(\{i\})$
p_{0j}: shorthand for $P_j(M)$
x_{ij}: 1 if j assigned to machine i, 0 otherwise
x_{0j}: 1 if j assigned globally, 0 otherwise

$$\min \; T \tag{IP-1}$$

$$\sum_{i=0}^{m} x_{ij} = 1 \qquad \text{for } j = 1, \ldots, n \tag{1}$$

$$\sum_{j=1}^{n} p_{ij} x_{ij} \leq T \qquad \text{for } i = 1, \ldots, m \tag{2}$$

$$\sum_{j=1}^{n} \sum_{i=0}^{m} p_{ij} x_{ij} \leq mT \tag{3}$$

$$p_{ij} x_{ij} \leq T \qquad \text{for } j = 1, \ldots, n \text{ and } i = 0, \ldots, m \tag{4}$$

ILP formulation for Hierarchical Scheduling

$p_{\alpha j}$: shorthand for $P_j(\alpha)$
$x_{\alpha j}$: 1 if j assigned affinity mask α, 0 otherwise

$$\min \; T \tag{IP-2}$$

$$\sum_{\alpha \in \mathcal{A}} x_{\alpha j} = 1 \qquad \text{for } j = 1, \ldots, n \tag{5}$$

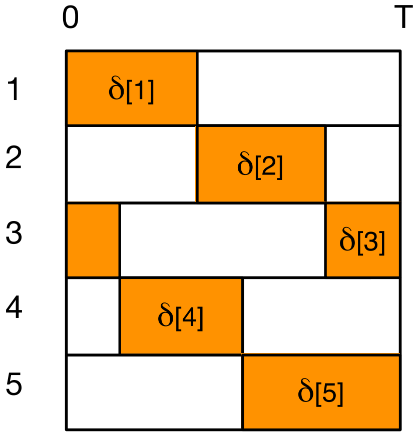
$$\sum_{j=1}^{n} \sum_{\beta \subseteq \alpha} p_{\beta j} x_{\beta j} \leq |\alpha| T \qquad \text{for each } \alpha \in \mathcal{A} \tag{6}$$

$$p_{\alpha j} x_{\alpha j} \leq T \qquad \text{for each } \alpha \in \mathcal{A} \text{ and } j = 1, \ldots, n \tag{7}$$

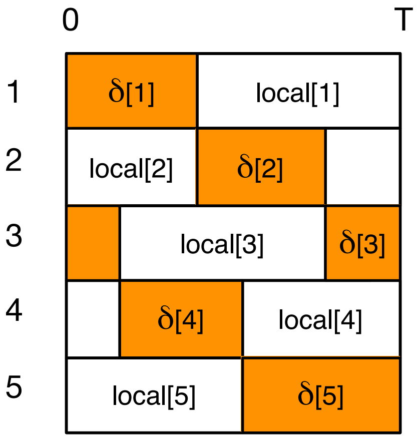
Constructing the schedule: Semi-Partitioned case

$t \leftarrow 0$; $i \leftarrow 0$;
$V \leftarrow \sum_{j=1}^{n} p_{0j} x_{0j}$;
while $V > 0$ **do**
  $i \leftarrow i + 1$;
  $\delta \leftarrow \min(V, T - \sum_{j=1}^{n} p_{ij} x_{ij})$;
  Assign δ units of global jobs to i, in the interval $[t, t + \delta \pmod{T})$];
  $t \leftarrow t + \delta \pmod{T}$;
  $V \leftarrow V - \delta$;
foreach machine $i \in M$ and job $j \in J$ such that $x_{ij} = 1$ **do**
  Schedule j on machine i in the idle time of interval $[0, T]$;

Example



Example



Constructing the schedule: Hierarchical case (sketch)

1. Bottom-up volume allocation phase:
   - Traverse $\mathcal{A}$ from local to global masks
   - Compute a "share" $\text{LOAD}[i, \alpha]$ of the jobs with mask α on machine i
   - Greedily assign the shares to more restricted machines first

Constructing the schedule: Hierarchical case (sketch)

1. Bottom-up volume allocation phase:
   - Traverse $\mathcal{A}$ from local to global masks
   - Compute a "share" $\textsc{Load}[i, \alpha]$ of the jobs with mask α on machine i
   - Greedily assign the shares to more restricted machines first
2. Top-down job assignment phase:
   - Traverse $\mathcal{A}$ from global to local masks
   - Use the share $\textsc{Load}[i, \alpha]$ to assign jobs with $x_{\alpha j} = 1$ to machine i

Constructing the schedule: Hierarchical case (sketch)

1. Bottom-up volume allocation phase:
   - Traverse $\mathcal{A}$ from local to global masks
   - Compute a “share” $\textsc{load}[i, \alpha]$ of the jobs with mask α on machine i
   - Greedily assign the shares to more restricted machines first
2. Top-down job assignment phase:
   - Traverse $\mathcal{A}$ from global to local masks
   - Use the share $\textsc{load}[i, \alpha]$ to assign jobs with $x_{\alpha j} = 1$ to machine i

Lemma

If $(\mathbf{x}, T)$ feasible for ILP, there exists a valid schedule with makespan T.

(In particular, no job is simultaneously scheduled on distinct machines)

Preprocessing the ILP

$p_{\alpha j}$: shorthand for $P_j(\alpha)$
$x_{\alpha j}$: 1 if j assigned affinity mask α, 0 otherwise

$$
\begin{aligned}
& \cancel{\min\ T} && && \text{(IP-2)} \\
& \sum_{\alpha \in \mathcal{A}} x_{\alpha j} = 1 && \text{for } j = 1, \dots, n && \\
& \sum_{j=1}^{n} \sum_{\beta \subseteq \alpha} p_{\beta j} x_{\beta j} \le |\alpha| T && \text{for each } \alpha \in \mathcal{A} && \\
& \cancel{p_{\alpha j} x_{\alpha j} \le T} && \cancel{\text{for each } \alpha \in \mathcal{A} \text{ and } j = 1, \dots, n} &&
\end{aligned}
$$

Preprocessing the ILP

$p_{\alpha j}$: shorthand for $P_j(\alpha)$
$x_{\alpha j}$: 1 if j assigned affinity mask α, 0 otherwise

$$\begin{aligned}
&\cancel{\min\ T} && && \text{(IP-2)}\\
&\sum_{\alpha \in \mathcal{A}} x_{\alpha j} = 1 && \text{for } j = 1, \ldots, n \\
&\sum_{j=1}^{n} \sum_{\beta \subseteq \alpha} p_{\beta j} x_{\beta j} \leq |\alpha| T && \text{for each } \alpha \in \mathcal{A} \\
&\cancel{p_{\alpha j} x_{\alpha j} \leq T} && \cancel{\text{for each } \alpha \in \mathcal{A} \text{ and } j = 1, \ldots, n}
\end{aligned}$$

Binary search for T

Preprocessing the ILP

$p_{\alpha j}$: shorthand for $P_j(\alpha)$
$x_{\alpha j}$: 1 if j assigned affinity mask α, 0 otherwise

$$\cancel{\min\ T} \tag{IP-2}$$

$$\sum_{\alpha \in \mathcal{A}} x_{\alpha j} = 1 \qquad \text{for } j = 1, \ldots, n$$

$$\sum_{j=1}^{n} \sum_{\beta \subseteq \alpha} p_{\beta j} x_{\beta j} \leq |\alpha| T \qquad \text{for each } \alpha \in \mathcal{A}$$

$$\cancel{p_{\alpha j} x_{\alpha j} \leq T} \qquad \cancel{\text{for each } \alpha \in \mathcal{A} \text{ and } j = 1, \ldots, n}$$

Binary search for T
Remove the $x_{\alpha j}$ with $p_{\alpha j} > T$

Preprocessing the ILP

$p_{\alpha j}$: shorthand for $P_j(\alpha)$
$x_{\alpha j}$: 1 if j assigned affinity mask α, 0 otherwise

$$\cancel{\min\ T} \tag{IP-2}$$

$$\sum_{\alpha \in \mathcal{A}} x_{\alpha j} = 1 \qquad \text{for } j = 1, \ldots, n$$

$$\sum_{j=1}^{n} \sum_{\beta \subseteq \alpha} p_{\beta j} x_{\beta j} \le |\alpha| T \qquad \text{for each } \alpha \in \mathcal{A}$$

$$\cancel{p_{\alpha j} x_{\alpha j} \le T} \qquad \cancel{\text{for each } \alpha \in \mathcal{A} \text{ and } j = 1, \ldots, n}$$

Binary search for T
Remove the $x_{\alpha j}$ with $p_{\alpha j} > T$
Decide either: (a) target T is infeasible, or (b) $2T$ is feasible

Fractional values “push-down” lemma

Lemma

Let $\mathbf{x}$ be LP-feasible, let $\beta \in \mathcal{A}$ ($|\beta| > 1$).
There exists another LP-feasible solution $\mathbf{x}'$ such that, for each job j,

$$x'_{\alpha j} = x_{\alpha j} \qquad \text{whenever } \alpha \not\subseteq \beta, \text{ and}$$
$$x'_{\beta j} = 0.$$

Fractional values "push-down" lemma

Lemma

Let $\mathbf{x}$ be LP-feasible, let $\beta \in \mathcal{A}$ ($|\beta| > 1$).
There exists another LP-feasible solution $\mathbf{x}'$ such that, for each job j,

$$x'_{\alpha j} = x_{\alpha j} \qquad \text{whenever } \alpha \not\subseteq \beta, \text{ and}$$
$$x'_{\beta j} = 0.$$

By repeated application, we remove all LP variables $x_{\alpha j}$ with $|\alpha| > 1$

Fractional values "push-down" lemma

Lemma

Let $\mathbf{x}$ be LP-feasible, let $\beta \in \mathcal{A}$ ($|\beta| > 1$).
There exists another LP-feasible solution $\mathbf{x}'$ such that, for each job j,

$$x'_{\alpha j} = x_{\alpha j} \qquad \text{whenever } \alpha \not\subseteq \beta, \text{ and}$$
$$x'_{\beta j} = 0.$$

By repeated application, we remove all LP variables $x_{\alpha j}$ with $|\alpha| > 1$

The LP becomes a standard unrelated machines LP

Fractional values "push-down" lemma

Lemma

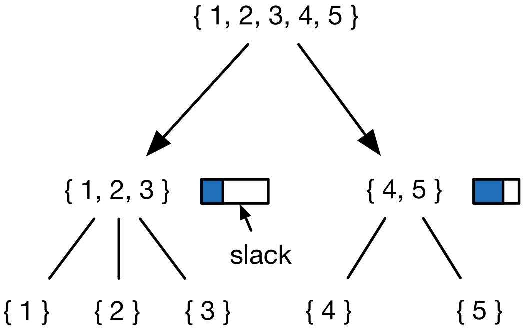
Let $\mathbf{x}$ be LP-feasible, let $\beta \in \mathcal{A}$ ($|\beta| > 1$).
There exists another LP-feasible solution $\mathbf{x}'$ such that, for each job j,

$$\begin{aligned} x'_{\alpha j} &= x_{\alpha j} && \text{whenever } \alpha \not\subseteq \beta, \text{ and} \\ x'_{\beta j} &= 0. \end{aligned}$$

By repeated application, we remove all LP variables $x_{\alpha j}$ with $|\alpha| > 1$

The LP becomes a standard unrelated machines LP
$\Rightarrow$ Invoke any LP-based rounding for $R||C_{\max}$ to obtain 2-approximation

Fractional values "push-down" lemma



Intuition: distribute value to children proportionally to their "slack":

$$\mathrm{slack}(\alpha) \stackrel{\mathrm{def}}{=} |\alpha| \cdot T - \sum_{j \in J} \sum_{\beta \subseteq \alpha} p_{\beta j} x_{\beta j}.$$

Summary and open problems

Main results

- A tractable scheduling model generalizing some well-studied problems
- Approximability result for the makespan objective with hierarchical affinity structure

Summary and open problems

Main results

- A tractable scheduling model generalizing some well-studied problems
- Approximability result for the makespan objective with hierarchical affinity structure

Open questions

- What if $\mathcal{A}$ is not hierarchical?
- Other objective functions
- Extensions of our rounding approach

Summary and open problems

Main results

- A tractable scheduling model generalizing some well-studied problems
- Approximability result for the makespan objective with hierarchical affinity structure

Open questions

- What if $\mathcal{A}$ is not hierarchical?
- Other objective functions
- Extensions of our rounding approach

THANKS!