Multiprocessor Speed Scaling with Precedence Constraints

Giorgio Lucarelli
LCOMS, Université de Lorraine

June 7, 2019

Joint work with Evripidis Bampis and Dimitrios Letsios

Outline

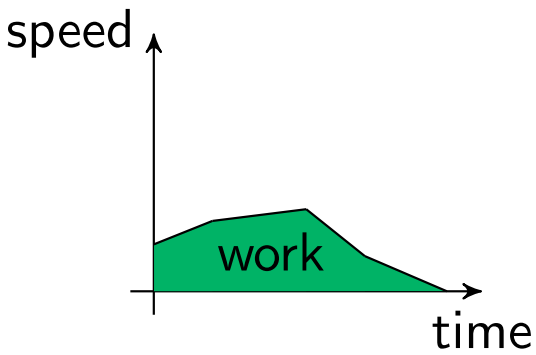
Speed Scaling

Speed Scaling:
Save energy by varying the processor's speed.

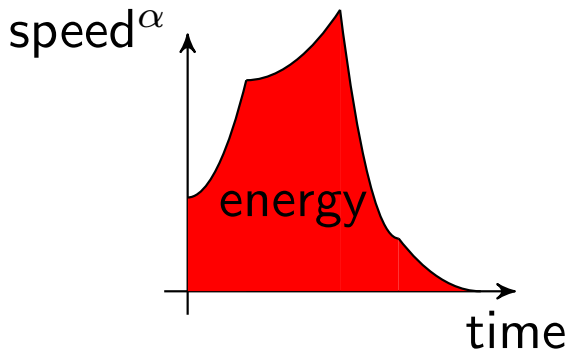
Model [Yao, Demers, Shenker, 1995]
Power consumption of a CMOS processor: $P(t) = s(t)^\alpha$

- CMOS: dominant technology for constructing microprocessors
- $s(t)$: processor's speed at time t
- $\alpha > 1$: machine-dependent constant, usually $\alpha \in (1, 3]$
- [Wierman, Andrew, Tang, 2009]:
  $\alpha = 1.11$ for Intel PXA 270, $\alpha = 1.62$ for Intel Pentium M 770

$w = \int s(t)dt$

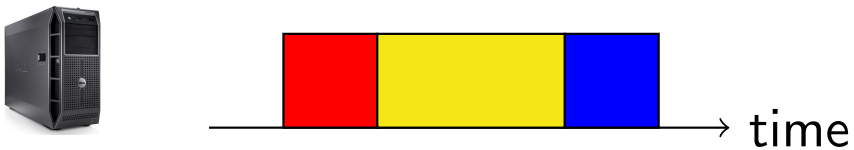


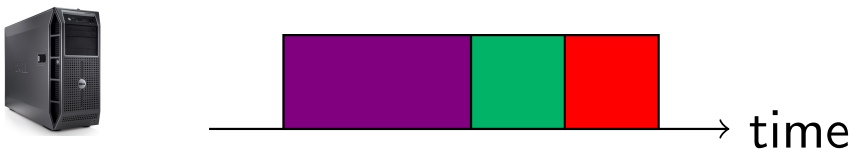
$E = \int P(t)dt$



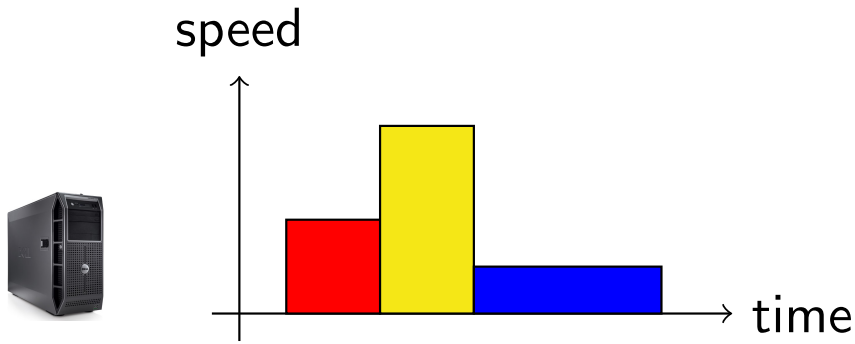
Scheduling with Speed Scaling

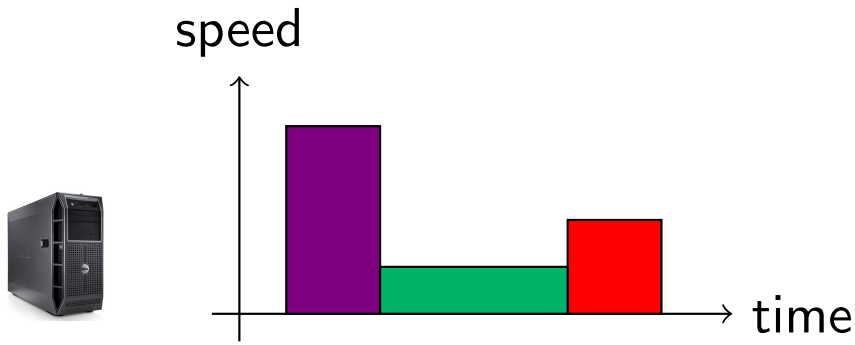
Scheduling:
Choose a running job every time.





Scheduling with Speed Scaling:
Decide the job and the speed.





Problem Definition

Instance:

- A set of n precedence-constrained jobs $\mathcal{J}$.
  - Job j has work w_j.
- A set of m parallel processors.
- A budget of energy E.

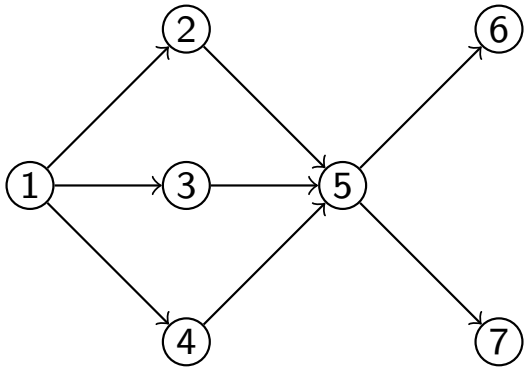
Objective:

- Find a non-preemptive schedule of minimum makespan, without exceeding the energy budget.

Precedence Constraints

Precedence constraints: directed acyclic graph $G = (V, E)$.

- V: a vertex for each job
- $(j, j') \in E$: j' can be executed only after the completion of j



Previous Work and Our Contribution

No Energy - Problem PREC:

- List Scheduling is $(2 - \frac{1}{m})$-approximate [GRAHAM, 1966].

Previous Work and Our Contribution

No Energy - Problem PREC:

- List Scheduling is $(2 - \frac{1}{m})$-approximate [GRAHAM, 1966].
- No $(2 - c)$-approximation algorithm where c is a constant, assuming a variant of the unique games conjecture [SVENSSON, 2010].

Previous Work and Our Contribution

No Energy - Problem PREC:

- List Scheduling is $(2 - \frac{1}{m})$-approximate [GRAHAM, 1966].
- No $(2 - c)$-approximation algorithm where c is a constant, assuming a variant of the unique games conjecture [SVENSSON, 2010].
- Uniformly Related processors (every machine i has speed s_i): $O(\log m)$-approximation algorithm [CHUDAK, SHMOYS 1999]

Previous Work and Our Contribution

No Energy - Problem PREC:

- List Scheduling is $(2 - \frac{1}{m})$-approximate [GRAHAM, 1966].
- No $(2 - c)$-approximation algorithm where c is a constant, assuming a variant of the unique games conjecture [SVENSSON, 2010].
- Uniformly Related processors (every machine i has speed s_i): $O(\log m)$-approximation algorithm [CHUDAK, SHMOYS 1999]

With Energy - Problem PREC_S:

- $O(\log^{1+2/\alpha} m)$-approximation algorithm [PRUHS ET AL. 2008]

Previous Work and Our Contribution

No Energy - Problem PREC:

- List Scheduling is $(2 - \frac{1}{m})$-approximate [GRAHAM, 1966].
- No $(2 - c)$-approximation algorithm where c is a constant, assuming a variant of the unique games conjecture [SVENSSON, 2010].
- Uniformly Related processors (every machine i has speed s_i): $O(\log m)$-approximation algorithm [CHUDAK, SHMOYS 1999]

With Energy - Problem PREC_S:

- $O(\log^{1+2/\alpha} m)$-approximation algorithm [PRUHS ET AL. 2008]
  1. Constant power schedules.
  2. Binary search to determine the power.
  3. Algorithm for uniformly related machines.

Previous Work and Our Contribution

No Energy - Problem PREC:

- List Scheduling is $(2 - \frac{1}{m})$-approximate [GRAHAM, 1966].
- No $(2 - c)$-approximation algorithm where c is a constant, assuming a variant of the unique games conjecture [SVENSSON, 2010].
- Uniformly Related processors (every machine i has speed s_i): $O(\log m)$-approximation algorithm [CHUDAK, SHMOYS 1999]

With Energy - Problem PREC_S:

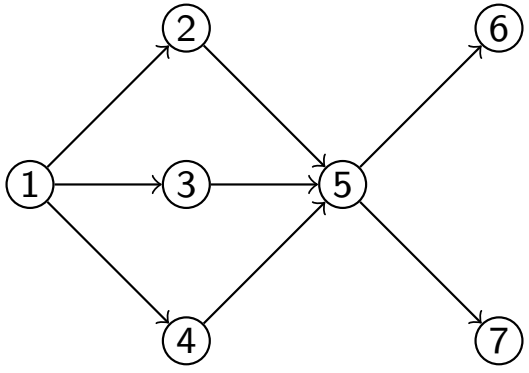
- $O(\log^{1+2/\alpha} m)$-approximation algorithm [PRUHS ET AL. 2008]
  1. Constant power schedules.
  2. Binary search to determine the power.
  3. Algorithm for uniformly related machines.
- $(2 - \frac{1}{m})$-approximation algorithm [this talk]

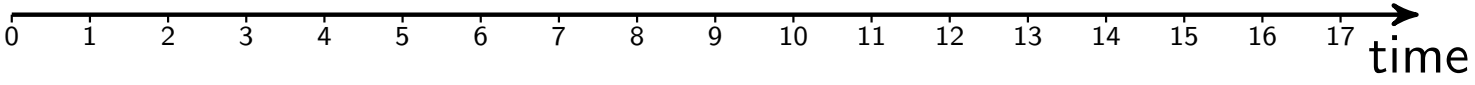
Outline

List Scheduling

List Scheduling (LS):

- Every time that a processor i becomes available, schedule on i a job of which all the predecessors have been completed.

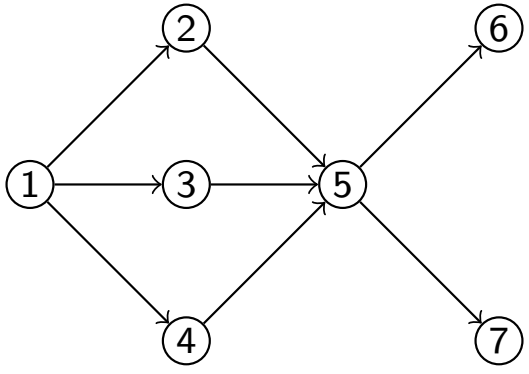


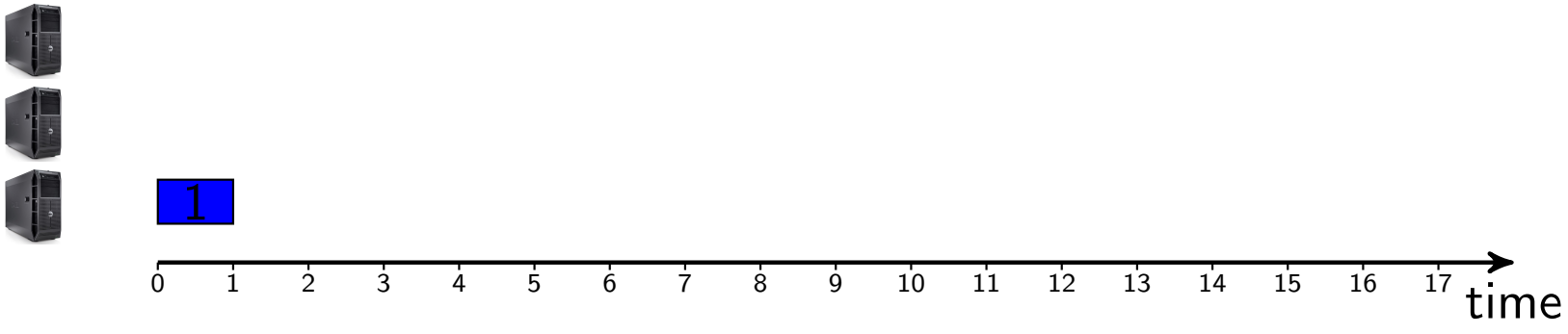


List Scheduling

List Scheduling (LS):

- Every time that a processor i becomes available, schedule on i a job of which all the predecessors have been completed.

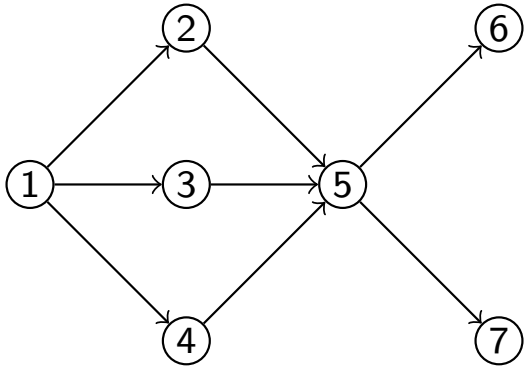


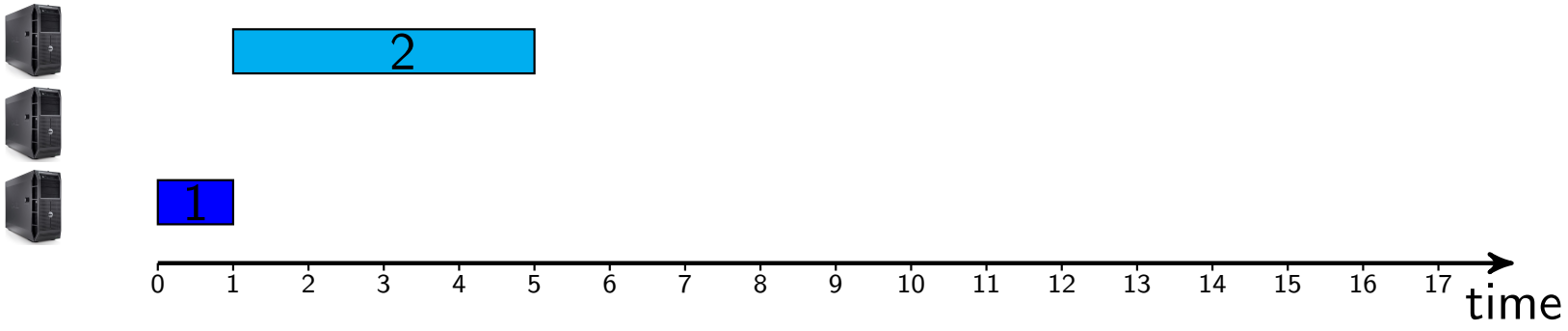


List Scheduling

List Scheduling (LS):

- Every time that a processor i becomes available, schedule on i a job of which all the predecessors have been completed.

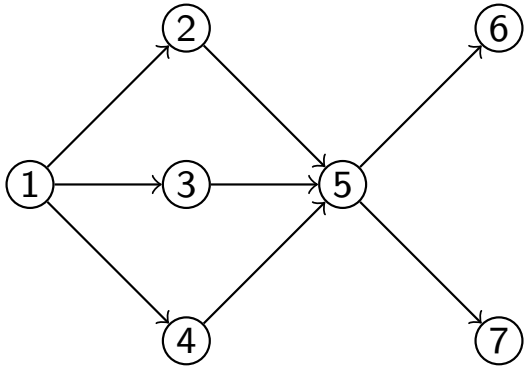


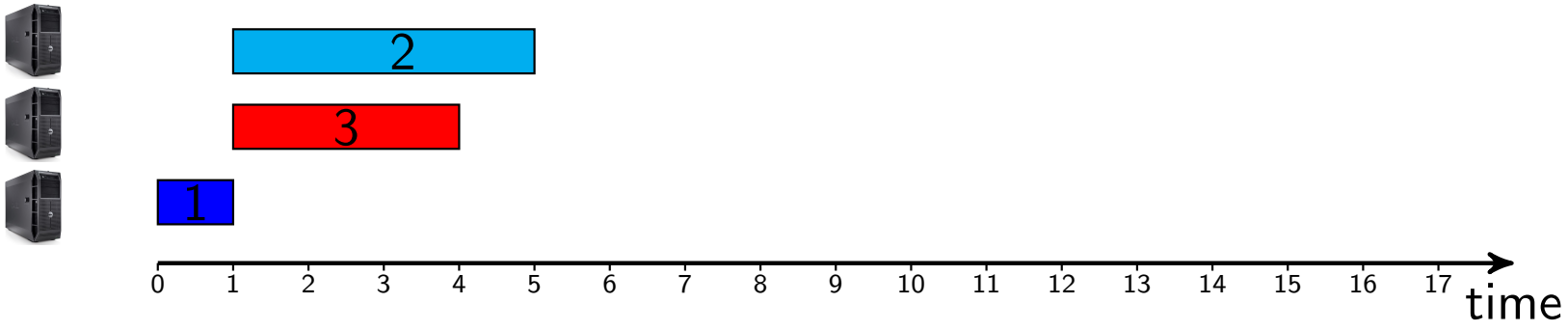


List Scheduling

List Scheduling (LS):

- Every time that a processor i becomes available, schedule on i a job of which all the predecessors have been completed.

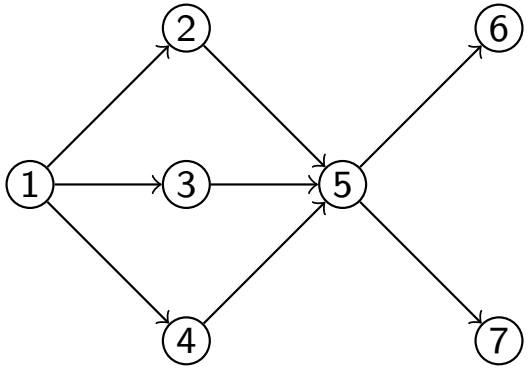


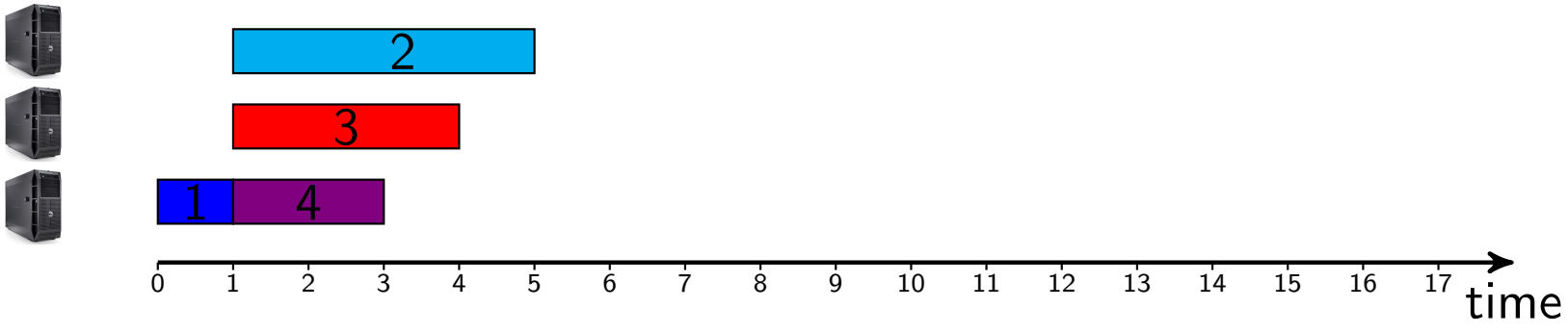


List Scheduling

List Scheduling (LS):

- Every time that a processor i becomes available, schedule on i a job of which all the predecessors have been completed.

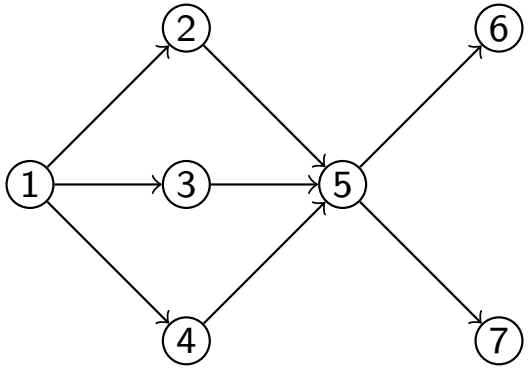


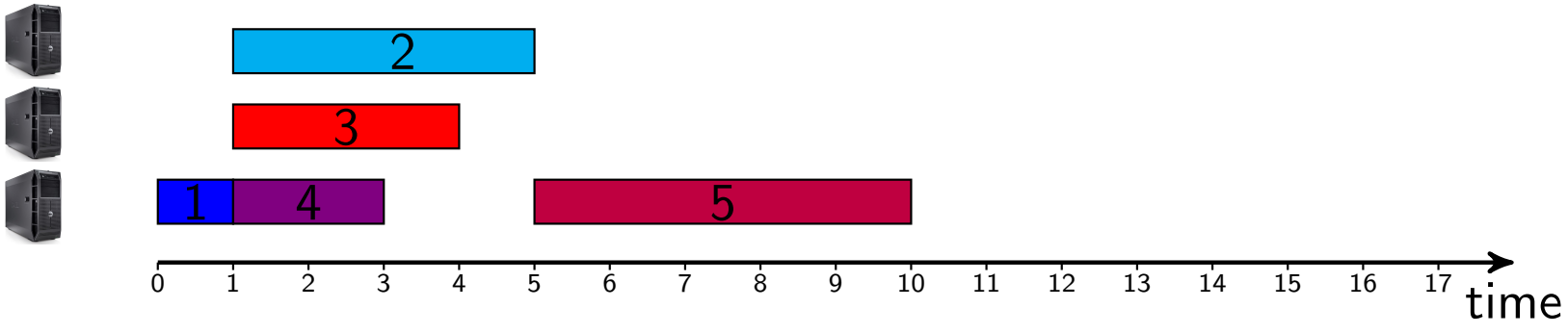


List Scheduling

List Scheduling (LS):

- Every time that a processor i becomes available, schedule on i a job of which all the predecessors have been completed.

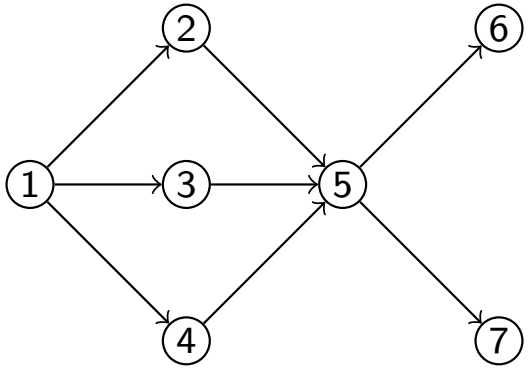


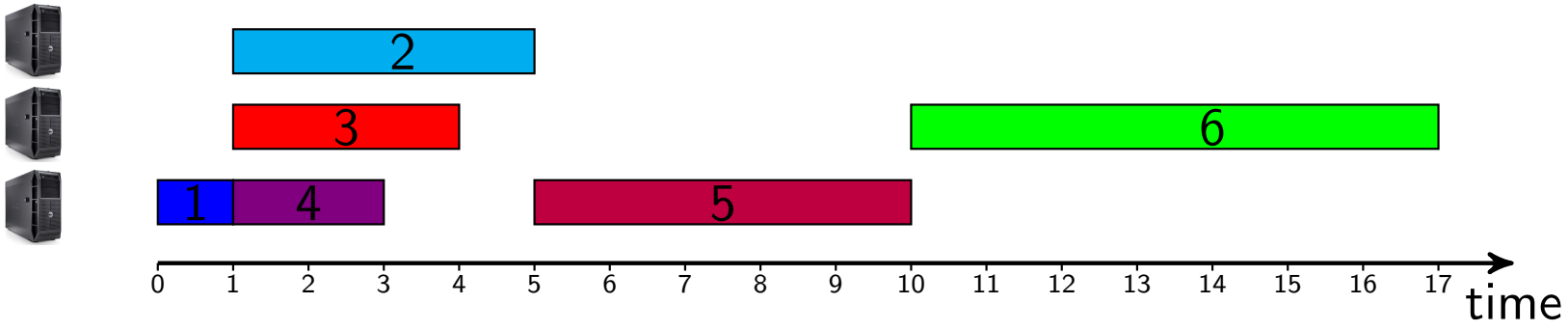


List Scheduling

List Scheduling (LS):

- Every time that a processor i becomes available, schedule on i a job of which all the predecessors have been completed.

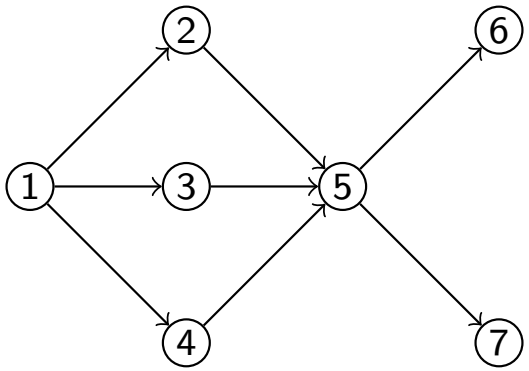


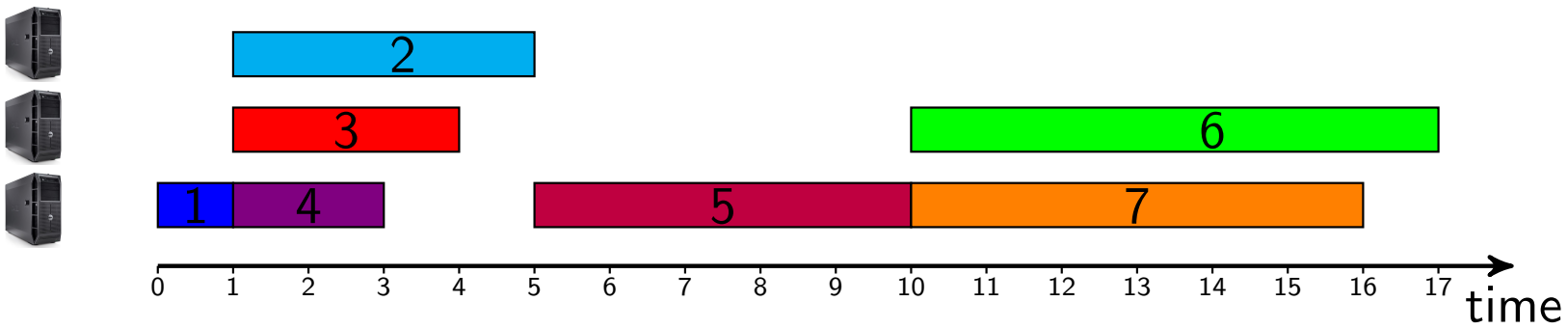


List Scheduling

List Scheduling (LS):

- Every time that a processor i becomes available, schedule on i a job of which all the predecessors have been completed.





List Scheduling

Theorem (GRAHAM, 1966)

LS is $(2 - \frac{1}{m})$-approximate for PREC.

Lower bounds on the makespan of the optimal schedule:

- $C_{opt} \geq \frac{1}{m} \sum_{j \in J} p_j$
- $C_{opt} \geq \sum_{j \in T} p_j$ for each path T of the graph G.

LS returns a schedule with makespan

$$C_{alg} \leq \left(2 - \frac{1}{m}\right) \max \left\{ \frac{1}{m} \sum_{j \in J} p_j, \max_{T} \left\{ \sum_{j \in T} p_j \right\} \right\}$$

Relation between Energy and Processing Times

Consequences of the convexity of the power function $P(s)$.

- In an optimal solution, each job j runs with constant speed s_j.
- Processing time of job j:

$$x_j = \frac{w_j}{s_j}$$

- Energy consumption for the execution of job j:

$$E_j = x_j \cdot s_j^\alpha = \frac{w_j^\alpha}{x_j^{\alpha-1}}$$

- The energy can be expressed as a convex function $E(\vec{x})$ of the vector $\vec{x}$ of the processing times of the jobs.

Convex Programming Relaxation

Variables:

- y: makespan
- x_j: processing time of job j

$$\min y$$

$$y \geq \frac{1}{m} \sum_{j \in J} x_j$$

$$y \geq \sum_{j \in T} x_j \quad \text{for every path } T \text{ of } G$$

$$\sum_{j \in J} \frac{w_j^\alpha}{x_j^{\alpha-1}} \leq E$$

$$x_j \geq 0$$

Convex Programming Relaxation

Variables:

- y: makespan
- x_j: processing time of job j

$$\min y$$

$$y \geq \frac{1}{m} \sum_{j \in J} x_j$$

$$y \geq \sum_{j \in T} x_j \quad \text{for every path } T \text{ of } G$$

$$\sum_{j \in J} \frac{w_j^\alpha}{x_j^{\alpha-1}} \leq E$$

$$x_j \geq 0$$

Remark:

- The convex program has an exponential number of constraints.

Approximation Algorithm

Algorithm:

- Compute an optimal solution $\vec{x}_{cp}$ of the convex program.
- Schedule the jobs by using List Scheduling and the processing times x_{cp}.

Theorem

The above algorithm is $(2 - \frac{1}{m})$*-approximate for* $PREC_S$.

Approximation Algorithm

Algorithm:

- Compute an optimal solution $\vec{x}_{cp}$ of the convex program.
- Schedule the jobs by using List Scheduling and the processing times x_{cp}.

Theorem

The above algorithm is $(2 - \frac{1}{m})$-approximate for $PREC_S$.

- $\vec{x}_{cp}$: processing times obtained by solving the convex program.
- $\vec{x}_{opt}$: processing times of an optimal schedule.
- $CP(\vec{x})$: value of the convex program w.r.t. the processing times $\vec{x}$.

$$C_{alg} \leq \left(2 - \frac{1}{m}\right) \cdot CP(\vec{x}_{cp}) \leq \left(2 - \frac{1}{m}\right) \cdot CP(\vec{x}_{opt}) \leq \left(2 - \frac{1}{m}\right) \cdot C_{opt}$$

A Convex Relaxation of Polynomial Size

The convex program has an exponential number of constraints:

$$y \geq \sum_{j \in T} x_j \quad \text{for every path } T \text{ of } G$$

We obtain an equivalent LP of polynomial size as follows:

- We introduce a variable y_j indicating the completion time of job j.
- We replace the above constraints with the following:

$$\begin{aligned} y_j &\leq y & j &\in J \\ x_j &\leq y_j & j &\in J \\ y_j + x_{j'} &\leq y_{j'} & (j, j') &\in E \\ y_j &\geq 0 & j &\in J \end{aligned}$$

So, we can solve the convex relaxation in polynomial time by applying the Ellipsoid algorithm to the new equivalent convex program.

Outline

Goal

Consider the following generic problems:

- Π: a classical makespan minimization problem
- Π_S: the speed scaling variant with a budget of energy

Assumption:
The energy consumption can be expressed as a convex function $E(\vec{x})$ of the vector of the processing times $\vec{x}$ of the jobs.

Is it possible to obtain a ρ-approximation algorithm for Π_S by using a known ρ-algorithm $\mathcal{A}$ for Π as a black box?

- Under some conditions, yes.

Conditions

- A set of ℓ linear bounds on Π's optimal solution of the form

$$C_{opt} \geq f_k(\vec{p}) \qquad k = 1, 2, \ldots, \ell$$

C_{opt}: value of the optimal solution for Π,
$\vec{p}$: the vector of processing times of the jobs,
$f_k(\vec{p})$: a linear function of $\vec{p}$.

- The ρ-approximation algorithm $\mathcal{A}$ always produces a solution for Π s.t.

$$C_{alg} \leq \rho \cdot \max_{k=1}^{\ell} \{f_k(\vec{p})\}$$

C_{alg}: value of the $\mathcal{A}$'s solution for Π.

Meta-Theorem

Theorem

The previous conditions imply a ρ-approximation algorithm for Π_S.

Algorithm:

1. Compute an optimal solution $\vec{x}_{cp}$ of the following convex program:

$$\begin{aligned} & \min y & \\ y &\geq f_k(\vec{x}) & k = 1, 2, \ldots, \ell \\ E(\vec{x}) &\leq E & \\ \vec{x} &\geq 0 & \end{aligned}$$

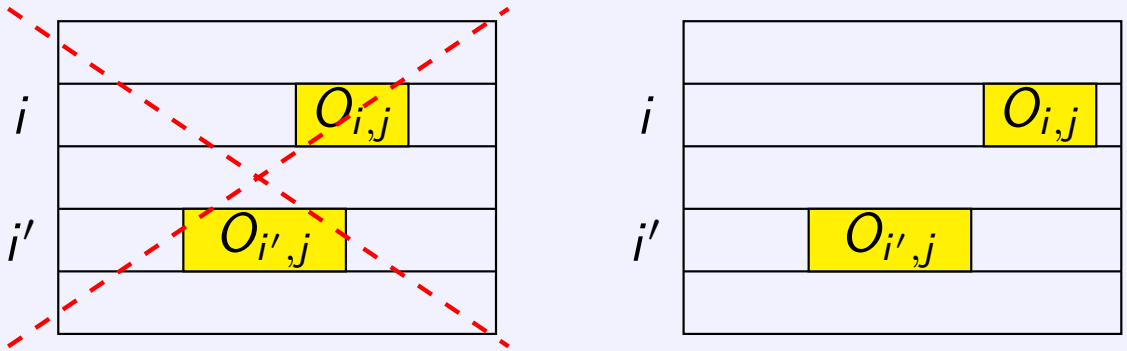
2. Schedule the jobs by using algorithm $\mathcal{A}$ and the processing times $\vec{x}_{cp}$.

Outline

Problem Definition

Instance:

- A set of n jobs J and a set of m processors P.
- Each job j consists of m operations $O_{1,j}, O_{2,j}, \ldots, O_{m,j}$.
  - Operation $O_{i,j}$ has an amount of work $w_{i,j}$ and it must be executed entirely by the processor i.
- Two operations of the same job cannot be executed at the same time.
- A budget of energy E.



Objective:

- Find a non-preemptive schedule of minimum makespan, without exceeding the energy budget.

Application of the Framework

Energy consumption: a convex function of the processing times of the operations.

Application of the Framework

Energy consumption: a convex function of the processing times of the operations.

Lower Bounds for problem SHOP without energy:

- $C_{opt} \geq \sum_{j \in \mathcal{J}} p_{i,j}$ for all $i \in P$.
- $C_{opt} \geq \sum_{i \in \mathcal{P}} p_{i,j}$ for all $j \in J$.

Application of the Framework

Energy consumption: a convex function of the processing times of the operations.

Lower Bounds for problem SHOP without energy:

- $C_{opt} \geq \sum_{j \in \mathcal{J}} p_{i,j}$ for all $i \in P$.
- $C_{opt} \geq \sum_{i \in \mathcal{P}} p_{i,j}$ for all $j \in J$.

List Scheduling (LS):
Whenever a processor $i \in \mathcal{P}$ becomes available, schedule on i an operation $O_{i,j}$ of a job $j \in \mathcal{J}$ which is not processed by any other processor at the same time.

[RACSMÁNY]:
LS produces a schedule for SHOP with makespan

$$C_{alg} \leq 2 \cdot \max \left\{ \max_{i \in \mathcal{P}} \left\{ \sum_{j \in \mathcal{J}} p_{i,j} \right\}, \max_{j \in \mathcal{J}} \left\{ \sum_{i \in \mathcal{P}} p_{i,j} \right\} \right\}$$

Application of the Framework

Energy consumption: a convex function of the processing times of the operations.

Lower Bounds for problem SHOP without energy:

- $C_{opt} \geq \sum_{j \in \mathcal{J}} p_{i,j}$ for all $i \in P$.
- $C_{opt} \geq \sum_{i \in \mathcal{P}} p_{i,j}$ for all $j \in J$.

List Scheduling (LS):
Whenever a processor $i \in \mathcal{P}$ becomes available, schedule on i an operation $O_{i,j}$ of a job $j \in \mathcal{J}$ which is not processed by any other processor at the same time.

[RACSMÁNY]:
LS produces a schedule for SHOP with makespan

$$C_{alg} \leq 2 \cdot \max \left\{ \max_{i \in \mathcal{P}} \left\{ \sum_{j \in \mathcal{J}} p_{i,j} \right\}, \max_{j \in \mathcal{J}} \left\{ \sum_{i \in \mathcal{P}} p_{i,j} \right\} \right\}$$

Therefore, there exists a 2-approximation algorithm for SHOP_S.

Outline

Concluding Remarks

Summary:

- $(2 - \frac{1}{m})$-approximation algorithm for the makespan minimization multiprocessor speed scaling problem with two steps:
  - Compute nice processing times for the jobs by solving a convex relaxation.
  - Apply list scheduling.
- A general framework for solving speed scaling problems.

Open questions:

- Is it possible to show "equivalence" between speed scaling and classical scheduling problems?
- Check the speed scaling version of the problem $1|pmtn, r_j|\sum C_j$.