Energy-aware algorithms

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr

http://graal.ens-lyon.fr/~abenoit

CR02 - 2016/2017

Exascale platforms

- Hierarchical
  - 10^5 or 10^6 nodes
  - Each node equipped with 10^4 or 10^3 cores
- Failure-prone

MTBF – one node	1 year	10 years	120 years
MTBF – platform of 10^6 nodes	30sec	5mn	1h

More nodes ⇒ Shorter MTBF (Mean Time Between Failures)

- Energy efficiency

  Thermal power close to the one of a nuclear reactor!
  A critical issue to address if we want to achieve Exascale.

Exascale platforms

- Hierarchical
  - 10^5 or 10^6 nodes
  - E...ch node equipped with 10^4 or 10^3 cores
- Failure-prone

MTBF – one node	...ar	10 years	120 years
MTBF – platform of 10^6 n...es	30s	5mn	1h

More nodes ⇒ Sh...er MTBF (Mean Tim... Between Failures)

- Energy eff...ency

Therm... po... ...a...or!
A c...cal is... ... Ex...cale.

Exascale
$\neq$ Petascale $\times 1000$

Outline

Energy: a crucial issue

- Data centers
  - 330, 000, 000, 000 Watts hour in 2007: more than France
  - 533, 000, 000 tons of CO_2: in the top ten countries
- Exascale computers (10^{18} floating operations per second)
  - Need effort for feasibility
  - 1% of power saved $\rightsquigarrow$ 1 million dollar per year
- Lambda user
  - 1 billion personal computers
  - 500, 000, 000, 000, 000 Watts hour per year
- $\rightsquigarrow$ crucial for both environmental and economical reasons

Energy: a crucial issue

- Data centers
  - 330, 000, 00 … ore than France
  - 533, 000, 00 … countries
- Exascale compu … ons per second)
  - Need effort
  - 1% of powe … r year
- Lambda user
  - 1 billion per
  - 500, 000, 00 … ear
- $\rightsquigarrow$ crucial for both environmental and economical reasons

Power dissipation of a processor

- $P = P_{\text{leak}} + P_{\text{dyn}}$
  - P_{leak}: constant
  - $P_{\text{dyn}} = B \times V^2 \times f$

    (B: constant; V: supply voltage; f: frequency)

- Standard approximation: $P = P_{\text{leak}} + f^{\alpha}$ $(2 \leq \alpha \leq 3)$
- Energy $E = P \times time$
- Dynamic Voltage and Frequency Scaling (DVFS) to reduce dynamic power
  - Real life: discrete speeds
  - Continuous speeds can be emulated
- Processor shutdown to reduce static power

Speed models for DVFS

		When can we change speed? Anytime	Beginning of tasks
Type of speeds	$[s_{\min}, s_{\max}]$	CONTINUOUS	-
	$\{s_1, ..., s_m\}$	VDD-HOPPING	DISCRETE, INCREMENTAL

- CONTINUOUS: great for theory
- Other "discrete" models more realistic
- VDD-HOPPING simulates CONTINUOUS
- INCREMENTAL is a special case of DISCRETE with equally-spaced speeds: for all $1 \leq q < m$, $s_{q+1} - s_q = \delta$

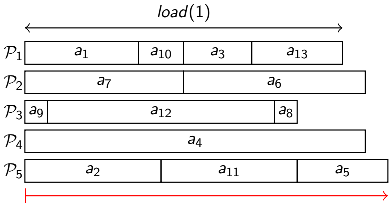
Outline

Framework

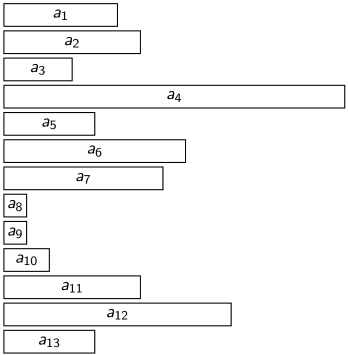
- Scheduling independent jobs
- GREEDY algorithm: assign next job to least-loaded processor
- Two variants:
  ONLINE-GREEDY: assign jobs on the fly
  OFFLINE-GREEDY: sort jobs before execution

Classical problem

- n independent jobs $\{J_i\}_{1 \le i \le n}$, a_i = size of J_i
- p processors $\{\mathcal{P}_q\}_{1 \le q \le p}$
- allocation function $alloc : \{J_i\} \rightarrow \{\mathcal{P}_q\}$
- load of $\mathcal{P}_q$ = $load(q) = \sum_{\{i \mid alloc(J_i) = \mathcal{P}_q\}} a_i$



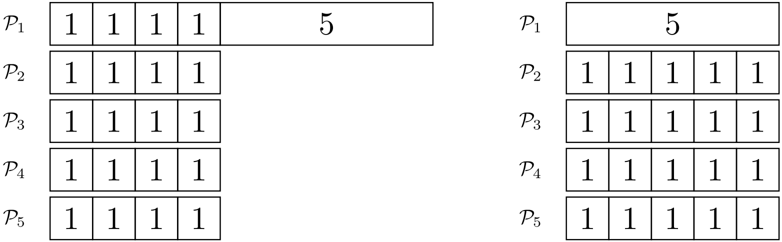
Execution time:
$\max_{1 \le q \le p} \quad load(q)$



OnLine-Greedy

Theorem

OnLine-Greedy *is a* $2 - \frac{1}{p}$ *approximation (tight bound)*

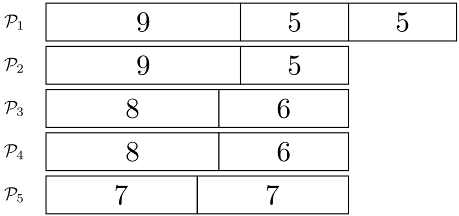


OnLine-Greedy — Optimal solution

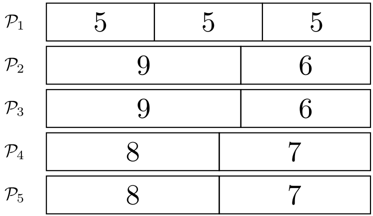
OffLine-Greedy

Theorem

OffLine-Greedy *is a* $\frac{4}{3} - \frac{1}{3p}$ *approximation (tight bound)*



OffLine-Greedy



Optimal solution

Power consumption

"The internet begins with coal"

- DVFS: Dynamic Voltage and Frequency Scaling
- Power at speed s (continuous model):

$$P(s) = P_{static} + \lambda \times s^3$$

Power consumption

Assess GREEDY for power!

"The internet begins with coal"

- DVFS: Dynamic Voltage and Frequency Scaling
- Power at speed s (continuous model):

$$P(s) = P_{static} + \lambda \times s^3$$

Outline

Bi-criteria problem

- Minimizing (dynamic) power consumption:
  $\Rightarrow$ use slowest possible speed $P_{dyn} = f^{\alpha} = f^3$

- Bi-criteria problem:
  Given bound $M = 1$ on execution time,
  minimize power consumption while meeting the bound

Bi-criteria problem statement

- n independent jobs $\{J_i\}_{1 \le i \le n}$, a_i = size of J_i
- p processors $\{\mathcal{P}_q\}_{1 \le q \le p}$
- allocation function $alloc : \{J_i\} \to \{\mathcal{P}_q\}$
- load of $\mathcal{P}_q$ = $load(q) = \sum_{\{i \mid alloc(J_i) = \mathcal{P}_q\}} a_i$

$(load(q))^3$ power dissipated by $\mathcal{P}_q$

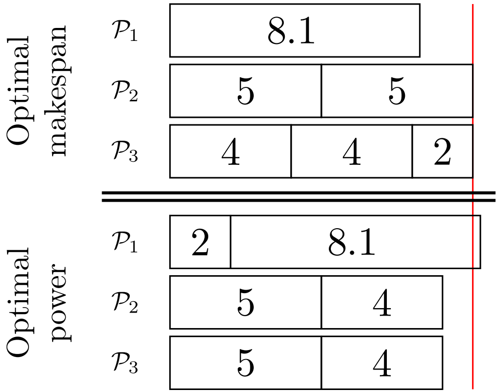
$\sum_{q=1}^{p} (load(q))^3$ **Power**

$\max_{1 \le q \le p} load(q)$ **Execution time**

Same GREEDY algorithm . . .

- Strategy: assign next job to least-loaded processor
- Natural for execution-time
  - smallest increment of maximum load
  - minimize objective value for currently processed jobs
- Natural for power too
  - smallest increment of total power (convexity)
  - minimize objective value for currently processed jobs

. . . but different optimal solution!



- Makespan 10, power 2531.441
- Makespan 10.1, power 2488.301

Outline

GREEDY and L_r norms

$$N_r = \left(\sum_{q=1}^{p} (load(q))^r \right)^{\frac{1}{r}}$$

- Execution time $N_\infty = \lim_{r \to \infty} N_r = \max_{1 \leq q \leq p} load(q)$
- Power $(N_3)^3$

Known results

N_2, OffLine-Greedy

- Chandra and Wong 1975: upper and lower bounds
- Leung and Wei 1995: tight approximation factor

N_3, OffLine-Greedy

- Chandra and Wong 1975: upper and lower bounds

N_r

- Alon et al. 1997: PTAS for offline problem
- Avidor et al. 1998: upper bound $2 - \Theta(\frac{\ln r}{r})$ for OnLine-Greedy

Contribution

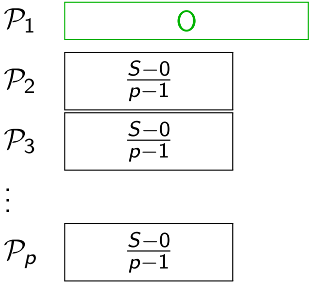
N_3

- Tight approximation factor for ONLINE-GREEDY
- Tight approximation factor for OFFLINE-GREEDY

- Greedy for power fully solved!

Outline

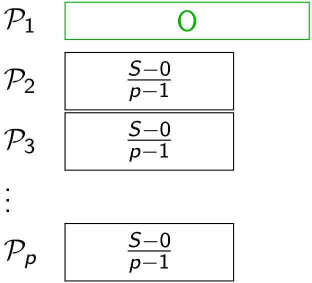
Best-case for optimal solution



O largest processor load in optimal solution, $S = \sum_{i=1}^{n} a_i$

$$Popt \geq O^3 + (p-1)\left(\frac{S-O}{p-1}\right)^3$$

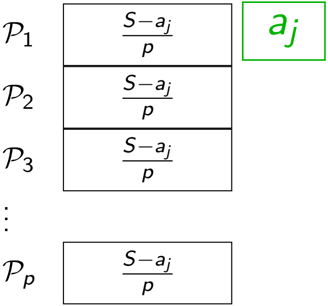
Best-case for optimal solution



O largest processor load in optimal solution, $S = \sum_{i=1}^{n} a_i$

$$Popt \geq O^3 + (p-1)\left(\frac{S-O}{p-1}\right)^3$$

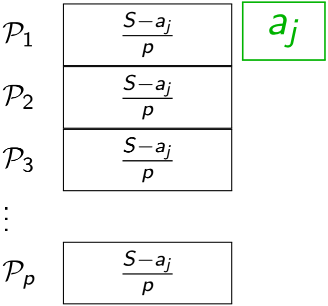
Worst-case for GREEDY



J_j last job assigned to most loaded processor in GREEDY

$$P_{\text{greedy}} \leq \left(\frac{S+(p-1)a_j}{p}\right)^3 + (p-1)\left(\frac{S-a_j}{p}\right)^3$$

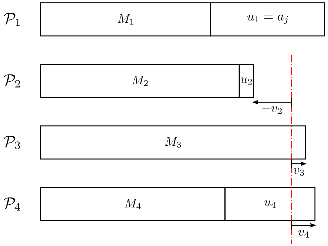
Worst-case for GREEDY



J_j last job assigned to most loaded processor in GREEDY

$$P_{\text{greedy}} \leq \left(\frac{S+(p-1)a_j}{p}\right)^3 + (p-1)\left(\frac{S-a_j}{p}\right)^3$$

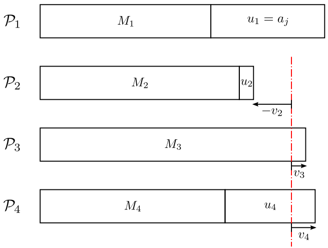
Proof sketch (1/3)



Notations

- $\mathcal{P}_1$ maximum loaded processor in GREEDY
- Load of of $\mathcal{P}_q$: M_q before job J_j, $M_q + u_q$ final
- $P_{\text{greedy}} = (M_1 + a_j)^3 + \sum_{q=2}^{p}(M_q + u_q)^3$

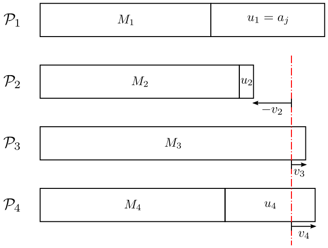
Proof sketch (2/3)



Notations

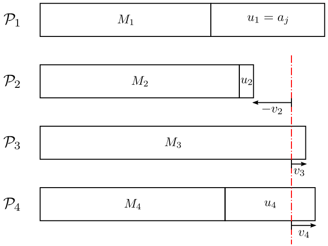
- For $q \geq 2$, rewrite $M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q$
- $P_{\text{greedy}} = \underbrace{(M_1 + a_j)^3 + \sum_{q=2}^{p} \left(\frac{S - M_1 - a_j}{p-1} + v_q \right)^3}_{f(M_1)}$

Proof sketch (3/3)



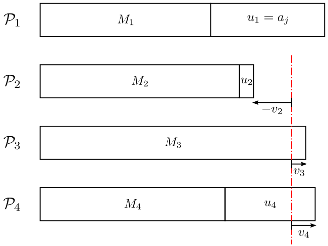
- Show: $f(M_1)$ strictly increasing
- Observe: $M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q$
- Derive: $M_1 \leq M_1^+ = \frac{S-a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
- Check: if $M_1 = M_1^+$, then $v_q = 0$ for all q
- Conclude ☺

Proof sketch (3/3)



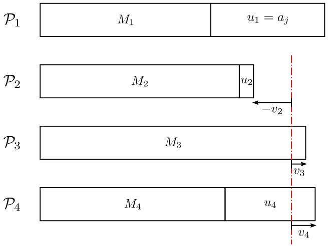
- Show: $f(M_1)$ strictly increasing
- Observe: $M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q$
- Derive: $M_1 \leq M_1^+ = \frac{S-a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
- Check: if $M_1 = M_1^+$, then $v_q = 0$ for all q
- Conclude ☺

Proof sketch (3/3)



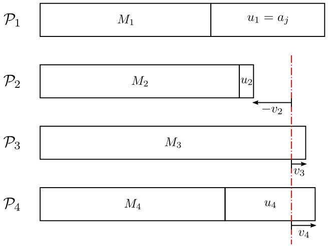
- Show: $f(M_1)$ strictly increasing
- Observe: $M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q$
- Derive: $M_1 \leq M_1^+ = \frac{S - a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
- Check: if $M_1 = M_1^+$, then $v_q = 0$ for all q
- Conclude ☺

Proof sketch (3/3)



- Show: $f(M_1)$ strictly increasing
- Observe: $M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q$
- Derive: $M_1 \leq M_1^+ = \frac{S - a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
- Check: if $M_1 = M_1^+$, then $v_q = 0$ for all q
- Conclude ☺

Proof sketch (3/3)



- Show: $f(M_1)$ strictly increasing
- Observe: $M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q$
- Derive: $M_1 \leq M_1^+ = \frac{S - a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
- Check: if $M_1 = M_1^+$, then $v_q = 0$ for all q
- Conclude ☺

Approximation bound

$$\frac{P_{\text{greedy}}}{P_{\text{opt}}} \leq \frac{\left(\frac{S+(p-1)a_j}{p}\right)^3 + (p-1)\left(\frac{S-a_j}{p}\right)^3}{O^3 + (p-1)\left(\frac{S-O}{p-1}\right)^3}$$

Agenda

- Right-hand-side is increasing with a_j
- Rewrite with $\beta = \frac{O}{S} \in [\frac{1}{p}, 1]$ and bound a_j:

$$\begin{array}{ll} a_j \leq O & \text{for OnLine-Greedy} \\ a_j \leq O/3 & \text{for OffLine-Greedy} \end{array}$$

Approximation for OnLine-Greedy

$$\frac{P_{\text{online}}}{P_{\text{opt}}} \leq \underbrace{\frac{\frac{1}{p^3}\left(\left(1+(p-1)\beta\right)^3+(p-1)\left(1-\beta\right)^3\right)}{\beta^3+\frac{(1-\beta)^3}{(p-1)^2}}}_{f_p^{(\text{on})}(\beta)}$$

Theorem

- *$f_p^{(\text{on})}$ has a single maximum in $\beta_p^{(\text{on})} \in [\frac{1}{p}, 1]$*
- OnLine-Greedy *is a $f_p^{(\text{on})}(\beta_p^{(\text{on})})$ approximation*
- *This approximation factor is tight*

Approximation for OffLine-Greedy

$$\frac{P_{\text{offline}}}{P_{\text{opt}}} \leq \underbrace{\frac{\frac{1}{p^3}\left(\left(1+\frac{(p-1)\beta}{3}\right)^3+(p-1)\left(1-\frac{\beta}{3}\right)^3\right)}{\beta^3+\frac{(1-\beta)^3}{(p-1)^2}}}_{f_p^{(\text{off})}(\beta)}$$

Theorem

- $f_p^{(\text{off})}$ *has a single maximum in* $\beta_p^{(\text{off})} \in [\frac{1}{p}, 1]$
- OffLine-Greedy *is a* $f_p^{(\text{off})}(\beta_p^{(\text{off})})$ *approximation*
- *This approximation factor is tight*

Numerical values of approximation ratios

p	Online-Greedy	Offline-Greedy
2	1.866	1.086
3	2.008	1.081
4	2.021	1.070
5	2.001	1.061
6	1.973	1.054
7	1.943	1.048
8	1.915	1.043
64	1.461	1.006
512	1.217	1.00083
2048	1.104	1.00010
2^{24}	1.006	1.000000025

Large values of p

Asymptotic approximation factors

OnLine-Greedy	$\frac{4}{3}$	1
OffLine-Greedy	2	1

↑ *optimal*

Conclusion

Contribution

- OnLine-Greedy and OffLine-Greedy for power
- Tight approximation factor for any p
- Extend long series of papers
- Completely solve N_3 minimization problem ☺

Extending to DAG workflows

- Reclaim the energy of existing list schedules
- Design (and assess) power-aware algorithms

Conclusion

Contribution

- OnLine-Greedy and OffLine-Greedy for power
- Tight approximation factor for any p
- Extend long series of papers
- Completely solve N_3 minimization problem ☺

Extending to DAG workflows

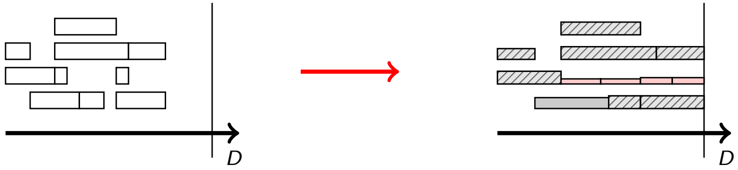
- Reclaim the energy of existing list schedules
- Design (and assess) power-aware algorithms

Outline

Motivation

- Mapping of tasks is given (ordered list for each processor and dependencies between tasks)
- If deadline not tight, why not take our time?
- Slack: unused time slots

Goal: efficiently use speed scaling (DVFS)



Outline

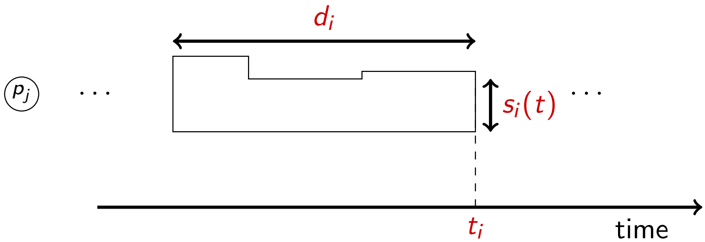
Speed models

		Change speed	
		Anytime	**Beginning of tasks**
Type of speeds	$[s_{\min}, s_{\max}]$	CONTINUOUS	-
	$\{s_1, ..., s_m\}$	VDD-HOPPING	DISCRETE, INCREMENTAL

- CONTINUOUS: great for theory (what we used for independent tasks!)
- Other "discrete" models more realistic
- VDD-HOPPING simulates CONTINUOUS
- INCREMENTAL is a special case of DISCRETE with equally-spaced speeds: for all $1 \leq q < m$, $s_{q+1} - s_q = \delta$

Tasks

- DAG: $\mathcal{G} = (V, E)$
- $n = |V|$ tasks T_i of weight $w_i = \int_{t_i - d_i}^{t_i} s_i(t)dt$
- d_i: task duration; t_i: time of end of execution of T_i



Parameters for T_i scheduled on processor p_j

Makespan

Assume T_i is executed at constant speed s_i

$$d_i = \mathcal{E}xe(w_i, s_i) = \frac{w_i}{s_i}$$

$$t_j + d_i \leq t_i \text{ for each } (T_j, T_i) \in E$$

Constraint on makespan:
$t_i \leq D$ for each $T_i \in V$

Energy

Energy to execute task T_i at speed s_i:

$$E_i(s_i) = d_i s_i^3 = w_i s_i^2$$

$\rightarrow$ Dynamic part of classical energy models

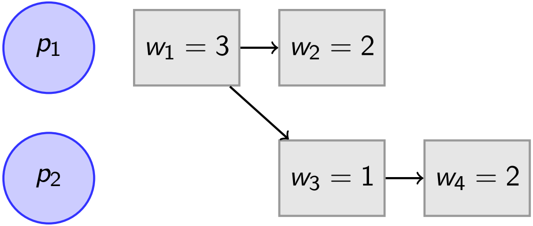
Bi-criteria problem

- Constraint on deadline: $t_i \leq D$ for each $T_i \in V$
- Minimize energy consumption: $\sum_{i=1}^{n} w_i \times s_i^2$

Outline

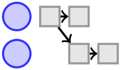
Example

Consider this DAG, with $s_{max} = 6$. Suppose deadline is $D = 1.5$.



Execution graph for the example.

Example



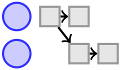
- CONTINUOUS: $(s_{max} = 6)$ $E_{opt}^{(c)} \simeq 109.6$.
  With the CONTINUOUS model, the optimal speeds are non rational values, and we obtain

$$s_1 = \frac{2}{3}(3 + 35^{1/3}) \simeq 4.18; \quad s_2 = s_1 \times \frac{2}{35^{1/3}} \simeq 2.56;$$

$$s_3 = s_4 = s_1 \times \frac{3}{35^{1/3}} \simeq 3.83.$$

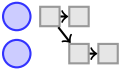
- DISCRETE: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E_{opt}^{(d)} = 170$.
- INCREMENTAL: $(\delta = 2,\ s_{\min} = 2,\ s_{\max} = 6)$ $E_{opt}^{(i)} = 128$.
- VDD-HOPPING: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E_{opt}^{(v)} = 144$.

Example



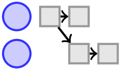
- CONTINUOUS: $(s_{max} = 6)$ $E_{opt}^{(c)} \simeq 109.6$.
- DISCRETE: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E_{opt}^{(d)} = 170$.
  For the DISCRETE model, if we execute all tasks at speed $s_2^{(d)} = 5$, we obtain an energy $E = 8 \times 5^2 = 200$. A better solution is obtained with $s_1 = s_3^{(d)} = 6$, $s_2 = s_3 = s_1^{(d)} = 2$ and $s_4 = s_2^{(d)} = 5$, which turns out to be optimal.
- INCREMENTAL: $(\delta = 2,\ s_{\min} = 2,\ s_{\max} = 6)$ $E_{opt}^{(i)} = 128$.
- VDD-HOPPING: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E_{opt}^{(v)} = 144$.

Example



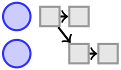
- CONTINUOUS: $(s_{max} = 6)$ $E_{opt}^{(c)} \simeq 109.6$.
- DISCRETE: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E_{opt}^{(d)} = 170$.
- INCREMENTAL: $(\delta = 2,\ s_{\min} = 2,\ s_{\max} = 6)$ $E_{opt}^{(i)} = 128$.
  For the INCREMENTAL model, the reasoning is similar to the DISCRETE case, and the optimal solution is obtained by an exhaustive search: all tasks should be executed at speed $s_2^{(i)} = 4$.
- VDD-HOPPING: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E_{opt}^{(v)} = 144$.

Example



- CONTINUOUS: $(s_{max} = 6)$ $E^{(c)}_{opt} \simeq 109.6$.
- DISCRETE: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E^{(d)}_{opt} = 170$.
- INCREMENTAL: $(\delta = 2,\ s_{\min} = 2,\ s_{\max} = 6)$ $E^{(i)}_{opt} = 128$.
- VDD-HOPPING: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E^{(v)}_{opt} = 144$.
  With the VDD-HOPPING model, we set $s_1 = s^{(d)}_2 = 5$; for the other tasks, we run part of the time at speed $s^{(d)}_2 = 5$, and part of the time at speed $s^{(d)}_1 = 2$ in order to use the idle time and lower the energy consumption.

Example



- CONTINUOUS: $(s_{max} = 6)$ $E_{opt}^{(c)} \simeq 109.6$.
- DISCRETE: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E_{opt}^{(d)} = 170$.
- INCREMENTAL: $(\delta = 2,\ s_{\min} = 2,\ s_{\max} = 6)$ $E_{opt}^{(i)} = 128$.
- VDD-HOPPING: $(s_1 = 2,\ s_2 = 5,\ s_3 = 6)$ $E_{opt}^{(v)} = 144$.

Outline

Complexity results

Minimizing energy with fixed mapping on p processors:

- CONTINUOUS: Polynomial for some special graphs, geometric optimization in the general case
- DISCRETE: NP-complete (reduction from 2-partition); approximation algorithm
- INCREMENTAL: NP-complete (reduction from 2-partition); approximation algorithm
- VDD-HOPPING: Polynomial (linear programming)

General problem: geometric programming

Reminder

For each task T_i,

- w_i is its size/work
- s_i is the speed of the processor that has task T_i assigned to
- t_i is the time when the computation of T_i ends

Objective function

$$\begin{aligned}
&\text{Minimize} \quad \textstyle\sum_{i=1}^{n} s_i^2 \times w_i \\
&\text{subject to (i) } t_i + \frac{w_j}{s_j} \le t_j \text{ for each } (T_i, T_j) \in E \\
&\qquad\qquad \text{(ii) } t_i \le D \text{ for each } T_i \in V
\end{aligned}$$

Results for continuous speeds

- MinEnergy(G,D) can be solved in polynomial time when G is a tree
- MinEnergy(G,D) can be solved in polynomial time when G is a series-parallel graph (assuming $s_{max} = +\infty$)

TODO: Prove the lemma for forks and joins to prove that MinEnergy(G,D) can be solved in polynomial time in this case (we just need to find s_0).