Algorithmes pour la minimisation de l'énergie

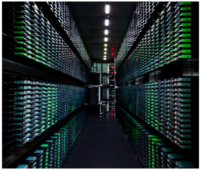
Giorgio Lucarelli

LIP6, Université Pierre et Marie Curie

JFRO, 8 Octobre 2013

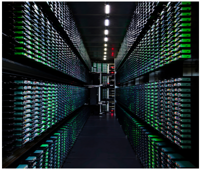
Energy-saving in computing systems

- Battery life of mobile devices
- Energy costs in data centers
- Temperature dissipation




Energy-saving in computing systems

- Battery life of mobile devices
- Energy costs in data centers
- Temperature dissipation

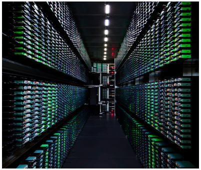



Solutions in

- Hardware
- Software

Energy-saving in computing systems

- Battery life of mobile devices
- Energy costs in data centers
- Temperature dissipation


Solutions in

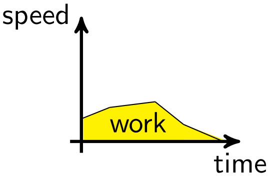
- Hardware
- Software → $\boxed{\textbf{Scheduling}}$

Speed scaling

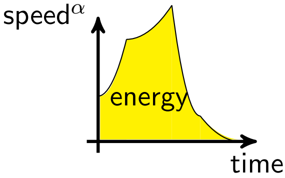
- $s(t)$: speed at time t (units of **work per** unit of **time**)
- $P(s(t)) = s(t)^\alpha$: power consumed by a CMOS device
  - CMOS: **dominant technology** for integrated circuits
  - $\alpha > 1$ is a machine-dependent constant
  - Intel PXA 270: 1.11, Intel Pentium M 770: 1.62
    [Wierman, Andrew, Tang; INFOCOM 2009]

- $s(t)$: speed at time t (units of **work per** unit of **time**)
- $P(s(t)) = s(t)^\alpha$: power consumed by a CMOS device
  - CMOS: **dominant technology** for integrated circuits
  - $\alpha > 1$ is a machine-dependent constant
  - Intel PXA 270: 1.11, Intel Pentium M 770: 1.62
    [Wierman, Andrew, Tang; INFOCOM 2009]

Work: $w = \int s(t)dt$



Energy: $E = \int P(s(t))dt$



The problem

Instance:

- A set of n jobs:
  - the job J_j has a work w_j, a release date r_j and a deadline d_j.
- Machine environment:
  - a single processor or a set of m parallel processors or a set of m heterogeneous processors or shop environments or ...

Objective:

- Find a feasible schedule of **minimum energy consumption**.

The problem

Instance:

- A set of n jobs:
  - the job J_j has a work w_j, a release date r_j and a deadline d_j.
- Machine environment:
  - a single processor or a set of m parallel processors or a set of m heterogeneous processors or shop environments or ...

Objective:

- Find a feasible schedule of **minimum energy consumption**.
  - We need to **determine the speed** of the processor(s).

The problem

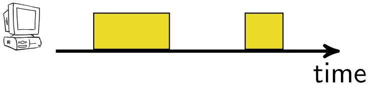
Instance:

- A set of n jobs:
  - the job J_j has a work w_j, a release date r_j and a deadline d_j.
- Machine environment:
  - a single processor or a set of m parallel processors or a set of m heterogeneous processors or shop environments or ...

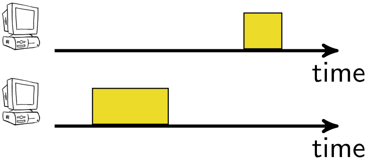
Objective:

- Find a feasible schedule of **minimum energy consumption**.
  - We need to **determine the speed** of the processor(s).

Preemption



Migration



Related work

	preemption		no-preemption
	migration	no-migration	
Single processor	polynomial [1]		NP-hard [2] $2^{5\alpha-4}$-approx. [2]
Parallel processors	polynomial [3,4,5]	NP-hard [6] B_α-approx. [7]	$m^\alpha(\sqrt[m]{n}^{\alpha-1})$-approx. [8]

[1. Yao, Demers, Shenker; FOCS 1995]

[2. Antoniadis, Huang; SWAT 2012]

[3. Albers, Antoniadis, Greiner; SPAA 2011]

[4. Angel, Bampis, Kacem, Letsios; EuroPar 2012]

[5. Bampis, Letsios, L.; ISAAC 2012]

[6. Albers, Müller, Schmelzer; SPAA 2007]

[7. Greiner, Nonner, Souza; SPAA 2009]

[8. Bampis, Kononov, Letsios, L., Nemparis; COCOON 2013]

Recent review: [Albers; STACS 2011]

Outline

- Linear programming and randomized rounding

  [Bampis, Kononov, Letsios, L., Sviridenko; FSTTCS 2013]

  - Heterogeneous multiprocessors without migrations

- Convex primal-dual

  [Bampis, Chau, Letsios, L., Milis; SEA 2013]

  - Open-shop with preemptions

Linear programming

and

Randomized rounding

Heterogeneity

- Each job J_j has
  - a different work w_{ij}
  - a different release date r_{ij}
  - a different deadline d_{ij}

  on each processor P_i.

- Each processor P_i has a different constant α_i.

- Each job J_j has
  - a different work w_{ij}
  - a different release date r_{ij}
  - a different deadline d_{ij}

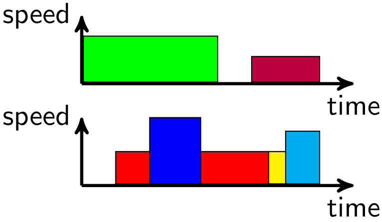
  on each processor P_i.

- Each processor P_i has a different constant α_i.

- Case study: we allow preemption but no migration of jobs

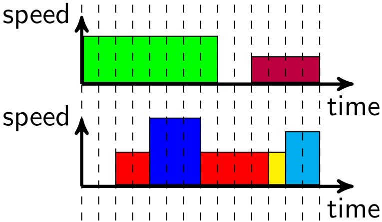
Integer programming formulation

Configuration: the schedule of a job



Integer programming formulation

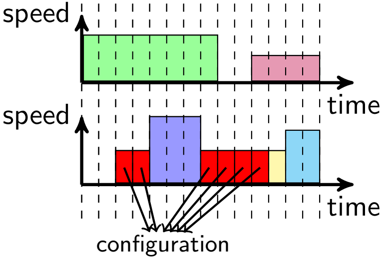
Configuration: the schedule of a job



- Discretize time
  - loose a factor of $1 + \epsilon$
  - polynomial to $1/\epsilon$ number of slots

Integer programming formulation

Configuration: the schedule of a job

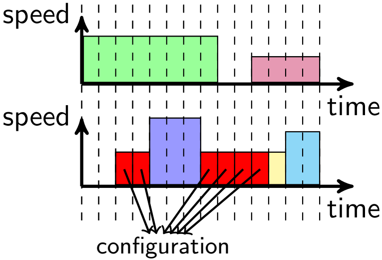


- Discretize time
  - loose a factor of $1+\epsilon$
  - polynomial to $1/\epsilon$ number of slots

Configuration: the set of slots of a job on a specific processor

Integer programming formulation

Configuration: the schedule of a job



- Discretize time
  - loose a factor of $1+\epsilon$
  - polynomial to $1/\epsilon$ number of slots

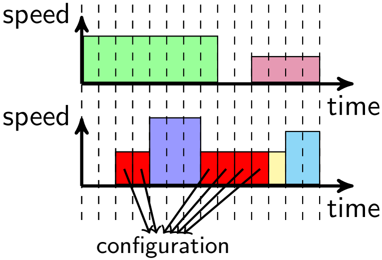
Configuration: the set of slots of a job on a specific processor

Given a configuration for the job J_j

- $s_{i,j,c}$: speed of J_j in configuration c on processor P_i
- $E_{i,j,c}$: energy consumption if J_j runs according to c on P_i

Integer programming formulation

Configuration: the schedule of a job



- Discretize time
  - loose a factor of $1+\epsilon$
  - polynomial to $1/\epsilon$ number of slots

Configuration: the set of slots of a job on a specific processor

Given a configuration for the job J_j

- $s_{i,j,c}$: speed of J_j in configuration c on processor P_i
- $E_{i,j,c}$: energy consumption if J_j runs according to c on P_i

$$x_{i,j,c} = \begin{cases} 1, & \text{if job } J_j \text{ is executed on } P_i \text{ according to } c \\ 0, & \text{otherwise} \end{cases}$$

Integer programming formulation

$$\min \sum_{i,j,c} E_{i,j,c} \cdot x_{i,j,c}$$

$$\sum_{i,c} x_{i,j,c} \geq 1 \quad \forall \text{ job } J_j$$

$$\sum_{s \in (i,j,c)} x_{i,j,c} \leq 1 \quad \forall \text{ slot } s$$

$$x_{i,j,c} \in \{0, 1\}$$

Integer programming formulation

$$\min \sum_{i,j,c} E_{i,j,c} \cdot x_{i,j,c}$$

$$\sum_{i,c} x_{i,j,c} \geq 1 \quad \forall \text{ job } J_j$$

$$\sum_{s \in (i,j,c)} x_{i,j,c} \leq 1 \quad \forall \text{ slot } s$$

$$x_{i,j,c} \geq 0$$

Integer programming formulation

$$\min \sum_{i,j,c} E_{i,j,c} \cdot x_{i,j,c}$$

$$\sum_{i,c} x_{i,j,c} \geq 1 \quad \forall \text{ job } J_j$$

$$\sum_{s \in (i,j,c)} x_{i,j,c} \leq 1 \quad \forall \text{ slot } s$$

$$x_{i,j,c} \geq 0$$

- #variables: exponential
- #constraints: polynomial

Integer programming formulation

$$\min \sum_{i,j,c} E_{i,j,c} \cdot x_{i,j,c}$$

$$\sum_{i,c} x_{i,j,c} \geq 1 \quad \forall \text{ job } J_j$$

$$\sum_{s \in (i,j,c)} x_{i,j,c} \leq 1 \quad \forall \text{ slot } s$$

$$x_{i,j,c} \geq 0$$

$$\max \sum_{j} \lambda_j - \sum_{s} \mu_s$$

$$\lambda_j - \sum_{s: s \in (i,j,c)} \mu_s \leq E_{i,j,c} \quad \forall (i,j,c)$$

$$\lambda_j, \mu_s \geq 0$$

- #variables: exponential
- #constraints: polynomial

- #variables: polynomial
- #constraints: exponential

Dual program

$$\max \sum_j \lambda_j - \sum_s \mu_s$$

$$\lambda_j - \sum_{s:s\in(i,j,c)} \mu_s \leq E_{i,j,c} \quad \forall (i,j,c)$$

$$\lambda_j, \mu_s \geq 0$$

Separation oracle:

- Given a solution (assignment to the variables)
  - either decides that the solution is feasible
  - or returns a violated constraint

Dual program

$$\max \sum_j \lambda_j - \sum_s \mu_s$$

$$\lambda_j - \sum_{s:s\in(i,j,c)} \mu_s \leq E_{i,j,c} \quad \forall (i,j,c)$$

$$\lambda_j, \mu_s \geq 0$$

Separation oracle:

- Given a solution (assignment to the variables)
  - either decides that the solution is feasible
  - or returns a violated constraint

- For each pair J_j and P_i find the configuration c that minimizes

$$E_{i,j,c} + \sum_{s:s\in(i,j,c)} \mu_s$$

Dual program

$$\max \sum_j \lambda_j - \sum_s \mu_s$$

$$\lambda_j - \sum_{s: s \in (i,j,c)} \mu_s \leq E_{i,j,c} \quad \forall (i,j,c)$$

$$\lambda_j, \mu_s \geq 0$$

Separation oracle:

- Given a solution (assignment to the variables)
  - either decides that the solution is feasible
  - or returns a violated constraint

- For each pair J_j and P_i find the configuration c that minimizes

$$E_{i,j,c} + \sum_{s: s \in (i,j,c)} \mu_s$$

- $E_{i,j,c}$: the same for configurations with equal number of slots
- For $x = 1, 2, \ldots, \#$slots, find the x variables μ_s with the minimum value

Solving the primal

Lemma ([Grötschel, Lovász, Schrijver; 1993])

The dual specifies a **polynomial number** *of violated constraints.*

- Solve the primal considering **only** the variables that correspond to violated constraints

Solving the primal

Lemma ([Grötschel, Lovász, Schrijver; 1993])

The dual specifies a **polynomial number** *of violated constraints.*

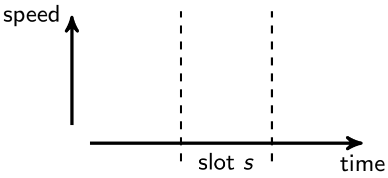
- Solve the primal considering **only** the variables that correspond to violated constraints

Theorem

We can find an optimal solution for the primal linear program in polynomial time.

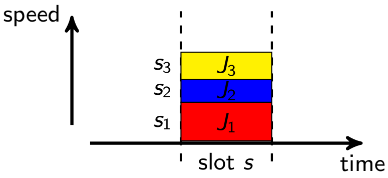
Randomized rounding

1. Solve the configuration LP relaxation.
2. For each job J_j, choose a configuration at random with probability $x_{i,j,c}$.
3. Scale the speeds during each slot such that to have a feasible schedule.



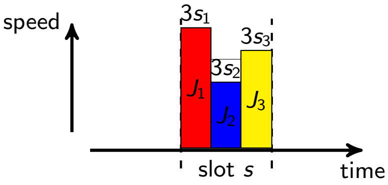
Randomized rounding

1. Solve the configuration LP relaxation.
2. For each job J_j, choose a configuration at random with probability $x_{i,j,c}$.
3. Scale the speeds during each slot such that to have a feasible schedule.



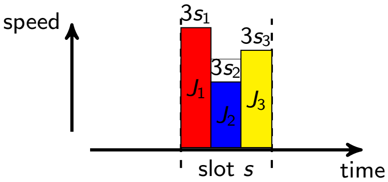
Randomized rounding

1. Solve the configuration LP relaxation.
2. For each job J_j, choose a configuration at random with probability $x_{i,j,c}$.
3. Scale the speeds during each slot such that to have a feasible schedule.



Randomized rounding

1. Solve the configuration LP relaxation.
2. For each job J_j, choose a configuration at random with probability $x_{i,j,c}$.
3. Scale the speeds during each slot such that to have a feasible schedule.

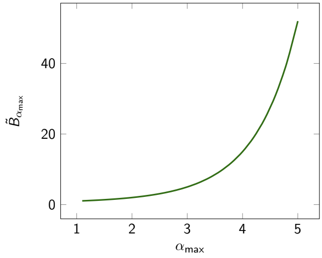


Theorem

The expectation of the energy consumption is no more than $\tilde{B}_{\alpha_{\max}}$ times the energy of the relaxed linear program.

Discussion

- $\tilde{B}_{\alpha_{\max}} = \sum_{k=0}^{\infty} \frac{k^{\alpha_{\max}}}{ek!}$
  - $\alpha_{\max}$-th (fractional) moment of Poisson's distribution
  - Intel PXA 270 : 1.067
  - Intel Pentium M 770 : 1.49
  - CMOS ($\alpha = 3$) : 5



Discussion

α	Preemptive without migrations		Non-preemptive single processor		Routing uniform demands	
	Homogeneous [1]	Heterogeneous [4]	[2]	[4]	[3]	[4]
1.11	2	$1.07(1+\varepsilon)$	2.93	$1.15(1+\varepsilon)$	375	1.07
1.62	2	$1.49(1+\varepsilon)$	17.15	$2.30(1+\varepsilon)$	2196	1.49
1.66	2	$1.54(1+\varepsilon)$	19.70	$2.43(1+\varepsilon)$	2522	1.54
2	2	$2(1+\varepsilon)$	64	$4(1+\varepsilon)$	8193	2
2.5	5	$3.08(1+\varepsilon)$	362	$8.72(1+\varepsilon)$	46342	3.08
3	5	$5(1+\varepsilon)$	2048	$20(1+\varepsilon)$	262145	5

[1. Greiner, Nonner, Souza; SPAA 2009]

[2. Antoniadis, Huang; SWAT 2012]

[3. Andrews, Anta, Zhang, Zhao; IEEE/ACM Trans. on Networking 2012]

[4. Bampis, Kononov, Letsios, L., Sviridenko; FSTTCS 2013]

- Heterogeneous multiprocessors with migrations
- Heterogeneous job-shop

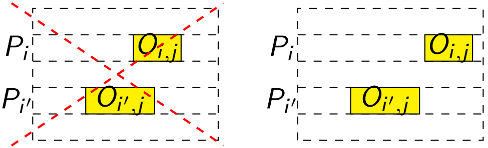
Convex primal-dual

Instance:

- A set of m parallel processors.
- A set of n jobs.
- Each job J_j has an operation O_{ij} with work $w_{ij} \geq 0$ to execute on the processor P_i.
- An available interval $[0, d]$.

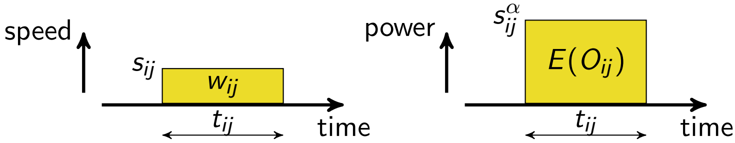
Objective:

- Find a feasible preemptive schedule with the minimum energy consumption such that operations of the same job are not executed in parallel.



Convexity

- each operation O_{ij} runs at constant speed $s_{ij} = \frac{w_{ij}}{t_{ij}}$
- $E(O_{ij}) = t_{ij} \cdot s_{ij}^{\alpha} = w_{ij} \cdot s_{ij}^{\alpha-1}$



The algorithm

1. Determine the speeds such that the total energy consumed is minimized
2. Transform works to processing times
3. Run the polynomial algorithm for the classical problem to determine the schedule

The algorithm

1. Determine the speeds such that the total energy consumed is minimized
2. Transform works to processing times
3. Run the polynomial algorithm for the classical problem to determine the schedule

The classical **preemptive openshop** problem

- Each operation O_{ij} has a processing time p_{ij} instead of work
- Polynomial-time algorithm that creates a feasible schedule [Gonzalez; IEEE Transactions on Computers 1979]

The algorithm

1. Determine the speeds such that the total energy consumed is minimized
2. Transform works to processing times
3. Run the polynomial algorithm for the classical problem to determine the schedule

Determine the speeds

- Convex cost flows [Bampis, Letsios, L.; ISAAC 2012]
- Convex program
- Convex primal-dual w.r.t. KKT conditions

Convex program

$$\min \sum_{O_{ij} \in J_j} \sum_{O_{ij} \in P_i} w_{ij} s_{ij}^{\alpha-1}$$

$$\sum_{O_{ij} \in P_i} \frac{w_{ij}}{s_{ij}} \leq d \qquad \text{for each } P_i$$

$$\sum_{O_{ij} \in J_j} \frac{w_{ij}}{s_{ij}} \leq d \qquad \text{for each } J_j$$

$$s_{ij} \geq 0 \qquad \text{for each } O_{ij}$$

- Necessary and sufficient conditions

Stationarity condition:

$$s_{ij}^{\alpha} = \frac{\beta_i + \gamma_j}{\alpha - 1} \quad \text{for each } O_{ij}$$

Complementary slackness conditions:

$$\beta_i \cdot \left(\sum_{O_{ij} \in P_i} \frac{w_{ij}}{s_{ij}} - d \right) = 0 \quad \text{for each } P_i$$

$$\gamma_j \cdot \left(\sum_{O_{ij} \in J_j} \frac{w_{ij}}{s_{ij}} - d \right) = 0 \quad \text{for each } J_j$$

Primal-dual method

Stationarity condition:

$$s_{ij}^{\alpha} = \frac{\beta_i + \gamma_j}{\alpha - 1} \quad \text{for each } O_{ij}$$

Complementary slackness conditions:

$$\beta_i \cdot \left(\sum_{O_{ij} \in P_i} \frac{w_{ij}}{s_{ij}} - d \right) = 0 \quad \text{for each } P_i$$

$$\gamma_j \cdot \left(\sum_{O_{ij} \in J_j} \frac{w_{ij}}{s_{ij}} - d \right) = 0 \quad \text{for each } J_j$$

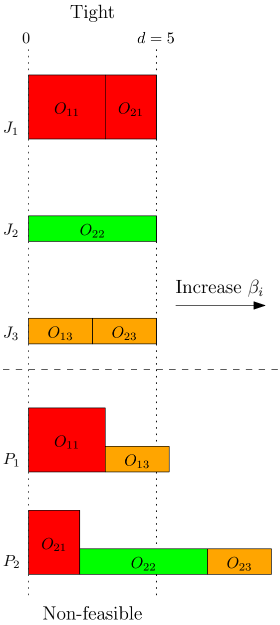
- Stationarity condition directly relates primal and dual variables
- **Main idea:** change the dual variables until complementary slackness conditions are satisfied

The primal-dual algorithm (an example)

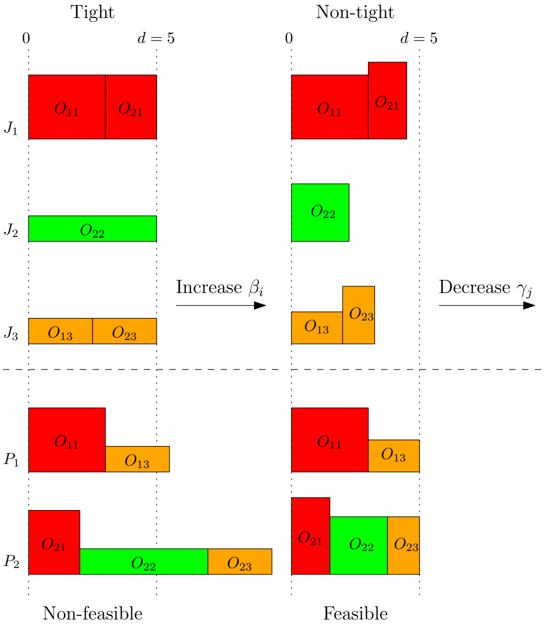
- Deadline $d = 5$
- Work

	J_1	J_2	J_3
P_1	3	-	1
P_2	2	2	1

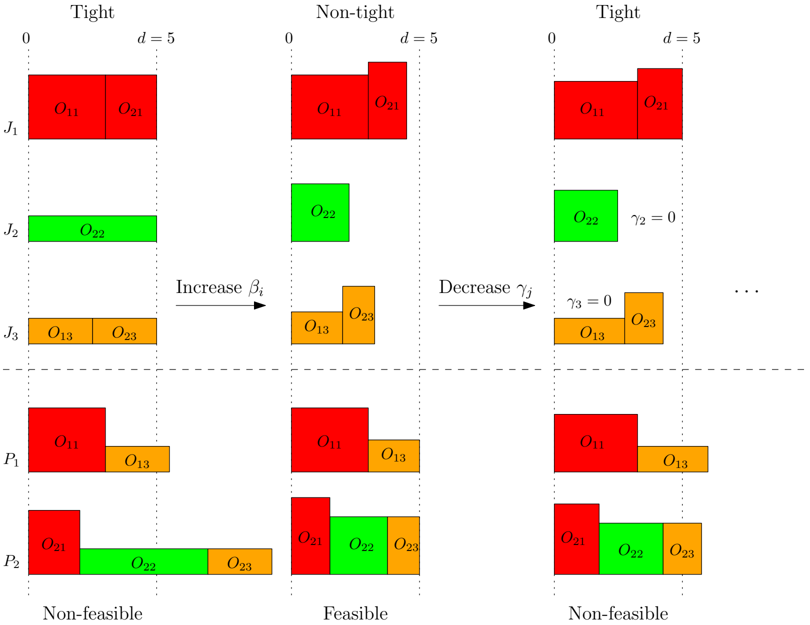
The primal-dual algorithm (an example)



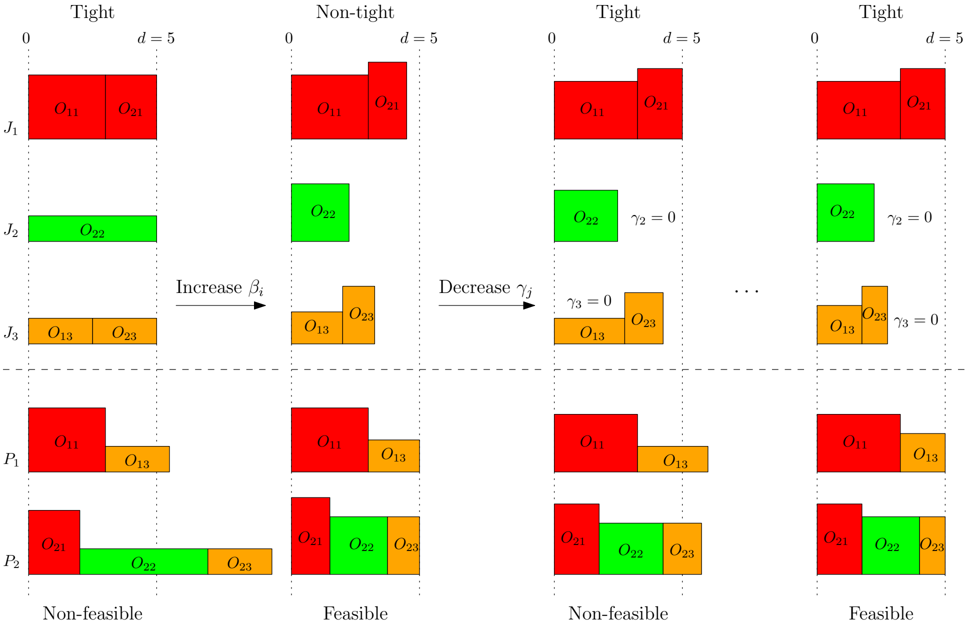
The primal-dual algorithm (an example)



The primal-dual algorithm (an example)



The primal-dual algorithm (an example)



The primal-dual algorithm

1. Initialize:
   - $\beta_i = 0$ and $\gamma_j = (\alpha - 1)\left(\frac{\sum_{O_{ij} \in J_j} w_{ij}}{d}\right)^{\alpha}$
2. While the complementary slackness conditions are not satisfied do
   1. **Increase** β_i to make **processors feasible**
   2. **Decrease** γ_j to make **jobs tight** or $\gamma_j = 0$

Our algorithm converges

- The algorithm converges, since at least one γ_j is decreased at each step
- **Complexity**?

- A: an array of size $m \times n$ with the work of operations
- $\alpha = 2$ or 2.5 or 3
- $d = 1000$

- $w_{\max} = 10$ or 50 or 100
- density: probability of an operation to exist
  $p = 0.5$ or 0.75 or 1

- 30 different instances for each combination of parameters

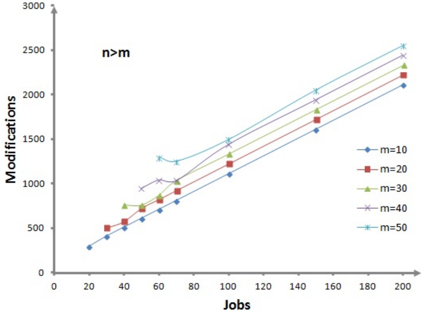
Number of modifications

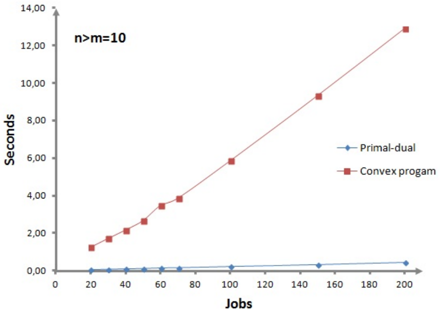
$\alpha = 2,\ w_{\max} = 10,\ p = 1$

n	$m = 5$	$m = 10$	$m = 15$	$m = 20$	$m = 25$	$m = 30$	$m = 40$	$m = 50$
5	40101	1	2	2	2	2	2	2
10	151	279611	3	4	3	4	4	4
20	255	295	384	–	34	7	7	10
30	355	410	443	500	593	–	12	15
40	455	510	565	572	640	756	–	32
50	555	610	665	720	768	755	947	–
60	655	710	765	820	872	864	1040	1294
70	755	810	865	920	975	1030	1034	1250
100	1055	1110	1165	1220	1275	1330	1440	1495
150	1555	1610	1665	1720	1775	1830	1940	2050
200	2055	2110	2165	2220	2275	2330	2440	2550

Case: $n > m$

$\alpha = 2,\ w_{\max} = 10,\ p = 1$



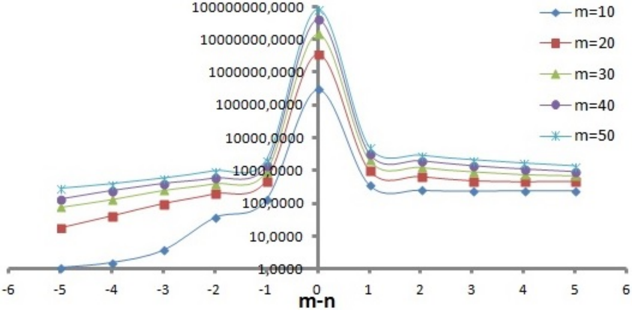


Case: $n = m = 10$

Parameters		Modifications
$\alpha = 2$, $w_{\text{max}} = 10$	$p = 0.5$	344
	$p = 0.75$	23915
	$p = 1$	179611
$w_{\text{max}} = 10$, $p = 1$	$\alpha = 2$	279611
	$\alpha = 2.5$	59785
	$\alpha = 3$	10716
$\alpha = 2$, $p = 1$	$w_{\text{max}} = 10$	279611
	$w_{\text{max}} = 50$	406608
	$w_{\text{max}} = 100$	–

Case: $n = m$

$\alpha = 2$, $w_{\max} = 10$, $p = 1$



Conclusions

Methodology

- Linear programming + Randomized rounding
- Convex programming + Primal dual

Conclusions

Methodology

- Linear programming + Randomized rounding
- Convex programming + Primal dual

Questions

- New models
- Tradeoffs between performance and energy

Thank you!