LATIN 2012

Low complexity scheduling algorithms minimizing the energy for tasks with agreeable deadlines

Eric Angel[1], Evripidis Bampis[2] and **Vincent Chau**[1]

[1]IBISC Laboratory; University of Évry – Val d'Essonne

[2]LIP6; University Pierre and Marie Curie

April, the 20^{th} 2012

Outline

Outline

Introduction

Results

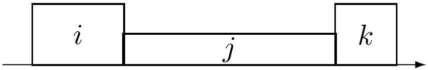
Conclusion

Motivation

- In 2010, server farm consumed 2% of all electricity
- 20 to 30% of servers are idle most of time
- We want to save the energy

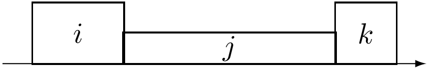
Two models

- Speed scaling
  - Power function : $\int_I s(t)^\alpha \mathrm{d}t$, for some $\alpha \in [2,3]$

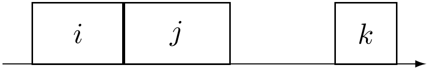


Two models

- Speed scaling
  - Power function : $\int_I s(t)^\alpha \mathrm{d}t$, for some $\alpha \in [2, 3]$



- Power-down mechanisms
  - No power consumption when idle
  - Cost L for wakeup



Our problem

- Power-down mechanisms
  - one sleep state and one active state
  - Transition from sleep to active state costs L

Our problem

- Power-down mechanisms
  - one sleep state and one active state
  - Transition from sleep to active state costs L
- We are given a set of n tasks : task j is characterized by
  - its release date r_j
  - its deadline d_j
  - its processing time p_j

Our problem

- Power-down mechanisms
  - one sleep state and one active state
  - Transition from sleep to active state costs L
- We are given a set of n tasks : task j is characterized by
  - its release date r_j
  - its deadline d_j
  - its processing time p_j
- Aim : Schedule all tasks minimizing the energy

Our problem

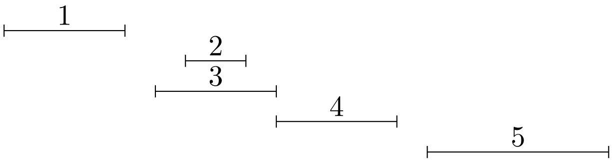
- Power-down mechanisms
  - one sleep state and one active state
  - Transition from sleep to active state costs L
- We are given a set of n tasks : task j is characterized by
  - its release date r_j
  - its deadline d_j
  - its processing time p_j
- Aim : Schedule all tasks minimizing the energy
- Cost for an idle period $[s, t)$ is $min\{L, t - s\}$

Our problem

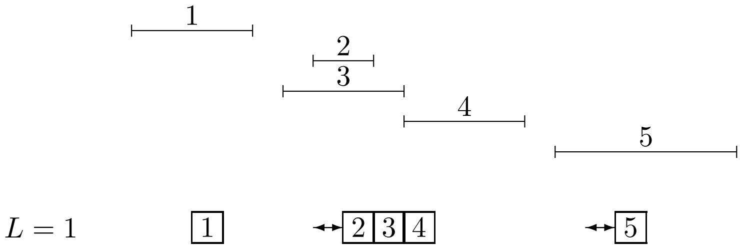
- Power-down mechanisms
  - one sleep state and one active state
  - Transition from sleep to active state costs L
- We are given a set of n tasks : task j is characterized by
  - its release date r_j
  - its deadline d_j
  - its processing time p_j
- Aim : Schedule all tasks minimizing the energy
- Cost for an idle period $[s, t)$ is $min\{L, t - s\}$

Remark : We focus on the energy's consumption of idle periods

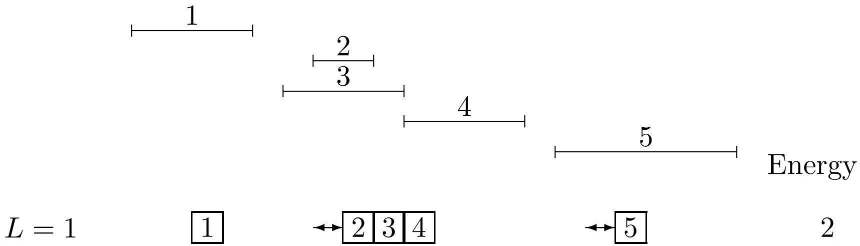
Example



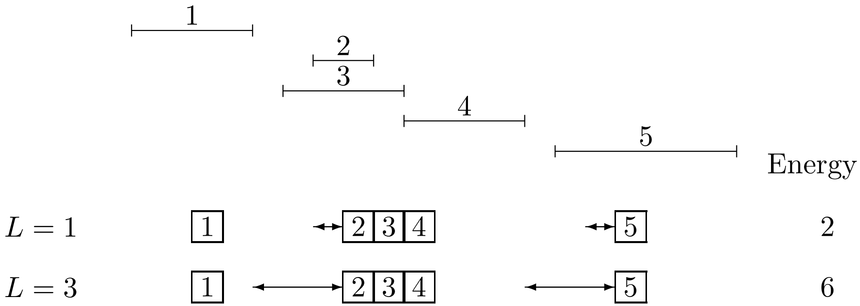
Example



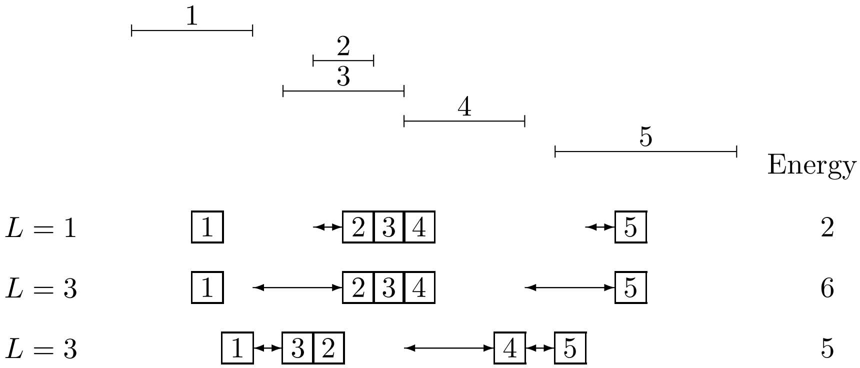
Example



Example



Example



What's known ?

# of processor	L	p_j	Assumption	Time
1	$\forall$	1		$O(n^7)$ [B'06] $O(n^4)$ [BCD'08]
1	$\forall$	$\forall$		$O(n^5)$ [BCD'08]
m	$\forall$	1		$O(n^7 m^5)$ [DGHSZ'07]
1	1	$\forall$	agreeable	$O(n \log n)$ [GJS'10]
1	$\forall$	1	agreeable	$O(n^3)$ [GJS'10] $O(n^2)$ **[ABC'12]**
1	$\forall$	$\forall$	agreeable	$O(n^2)$ **[ABC'12]**
m	1	1	agreeable	$O(n^3 m^2)$ [GJS'10] $O(n^2 m)$ **[ABC'12]**

[B'06] = Baptiste

[BCD'08] = Baptiste, Chrobak, Dürr

[DGHSZ'07] = Demaine, Ghodsi, Hajiaghayi, Sayedi-Roshkhar, Zadimoghaddam

[GJS'10] = Gururaj, Jalan, Stein

[ABC'12] = Angel, Bampis, Chau

Outline

Monoprocessor case, agreeable

Tasks have *agreeable deadlines* : for every pair of tasks i and j, one has $r_i \leq r_j$ if and only if $d_i \leq d_j$

Monoprocessor case, agreeable

Tasks have *agreeable deadlines* : for every pair of tasks i and j, one has $r_i \leq r_j$ if and only if $d_i \leq d_j$

Proposition 1

There exists an optimal solution in which all tasks are scheduled according to the EDF order.

Monoprocessor case, agreeable

Tasks have *agreeable deadlines* : for every pair of tasks i and j, one has $r_i \leq r_j$ if and only if $d_i \leq d_j$

Proposition 1

There exists an optimal solution in which all tasks are scheduled according to the EDF order.

Proposition 2

Without loss of generality, we may assume that $d_i \leq d_{i+1} - p_{i+1}$ and $r_j + p_j \leq r_{j+1}$

Monoprocessor case, agreeable

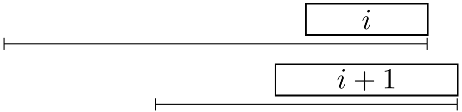
Tasks have *agreeable deadlines* : for every pair of tasks i and j, one has $r_i \leq r_j$ if and only if $d_i \leq d_j$

Proposition 1

There exists an optimal solution in which all tasks are scheduled according to the EDF order.

Proposition 2

Without loss of generality, we may assume that $d_i \leq d_{i+1} - p_{i+1}$ and $r_j + p_j \leq r_{j+1}$



Monoprocessor case, agreeable

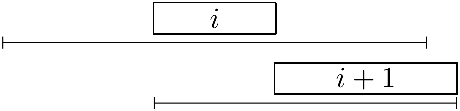
Tasks have *agreeable deadlines* : for every pair of tasks i and j, one has $r_i \leq r_j$ if and only if $d_i \leq d_j$

Proposition 1

There exists an optimal solution in which all tasks are scheduled according to the EDF order.

Proposition 2

Without loss of generality, we may assume that $d_i \leq d_{i+1} - p_{i+1}$ and $r_j + p_j \leq r_{j+1}$



Monoprocessor case, agreeable

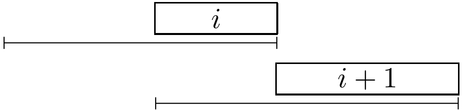
Tasks have *agreeable deadlines* : for every pair of tasks i and j, one has $r_i \leq r_j$ if and only if $d_i \leq d_j$

Proposition 1

There exists an optimal solution in which all tasks are scheduled according to the EDF order.

Proposition 2

Without loss of generality, we may assume that $d_i \leq d_{i+1} - p_{i+1}$ and $r_j + p_j \leq r_{j+1}$



Monoprocessor case, agreeable

Proposition 3

There exists an optimal solution in which all tasks are scheduled according to the EDF order, and such that for any task i, either task i is scheduled at

P1) its release date r_i, or

P2) the completion time of task $i-1$, or

P3) time $d_i - p_i$.

Monoprocessor case, agreeable

Proposition 3

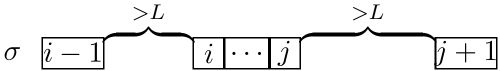
There exists an optimal solution in which all tasks are scheduled according to the EDF order, and such that for any task i, either task i is scheduled at

P1) its release date r_i, or

P2) the completion time of task $i-1$, or

P3) time $d_i - p_i$.

Proof idea :



Monoprocessor case, agreeable

Proposition 3

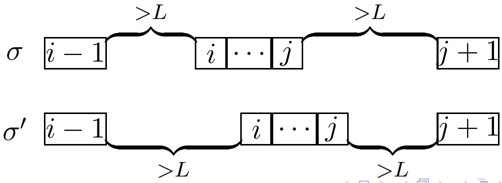
There exists an optimal solution in which all tasks are scheduled according to the EDF order, and such that for any task i, either task i is scheduled at

P1) its release date r_i, or

P2) the completion time of task $i-1$, or

P3) time $d_i - p_i$.

Proof idea :



Calcul

According to Proposition 3, we can define :

D_k : Set of possible completion times of task k
$D_1 = \{d_1\}$
$D_k = \bigcup_{t \in D_{k-1}} \{t + p_k | t \geq r_k\} \cup \{r_k + p_k\} \cup \{d_k\}$

Calcul

According to Proposition 3, we can define :

D_k : Set of possible completion times of task k
$D_1 = \{d_1\}$
$D_k = \bigcup_{t \in D_{k-1}} \{t + p_k | t \geq r_k\} \cup \{r_k + p_k\} \cup \{d_k\}$

$E_k(t)$: Energy's consumption (of idle periods) of the k first tasks such that task k is scheduled at time $t - p_k$

Calcul

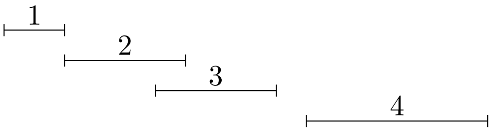
According to Proposition 3, we can define :

D_k : Set of possible completion times of task k
$D_1 = \{d_1\}$
$D_k = \bigcup_{t \in D_{k-1}} \{t + p_k | t \geq r_k\} \cup \{r_k + p_k\} \cup \{d_k\}$

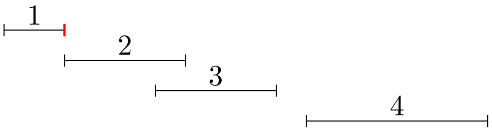
$E_k(t)$: Energy's consumption (of idle periods) of the k first tasks such that task k is scheduled at time $t - p_k$

Objective function : $\min_{t \in D_n} E_n(t)$

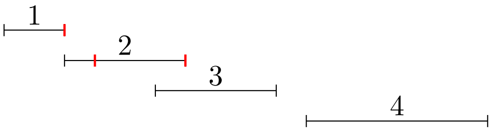
Example



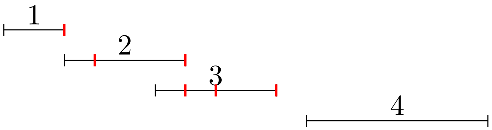
Example



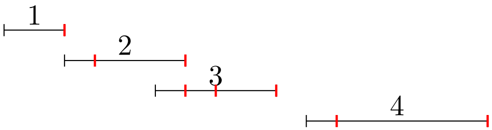
Example



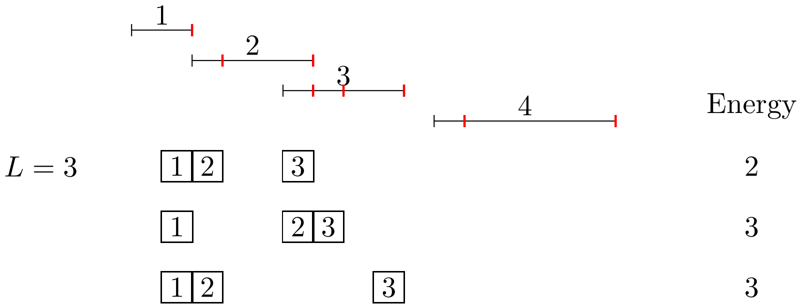
Example



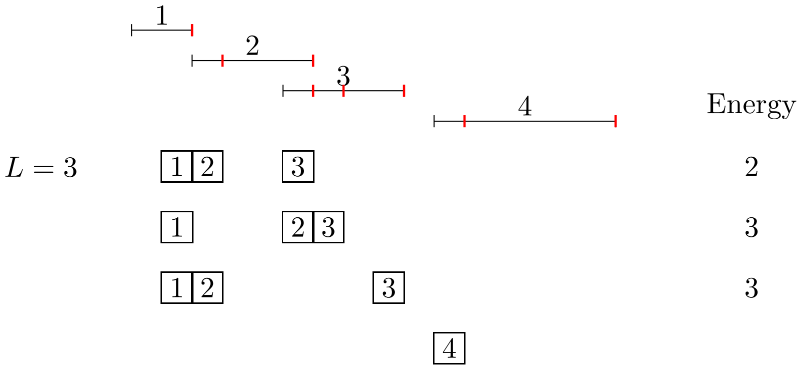
Example



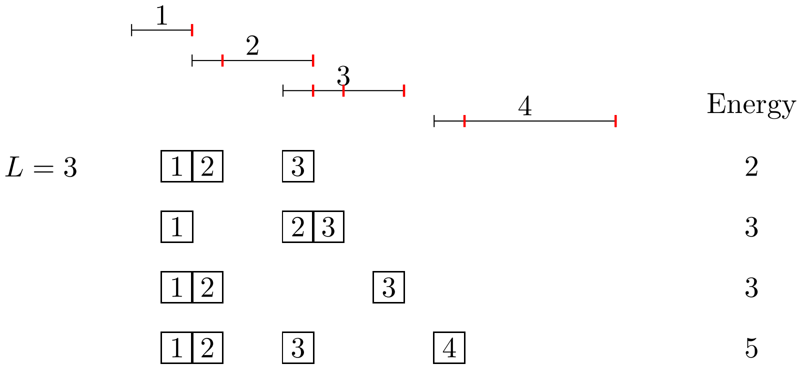
Example



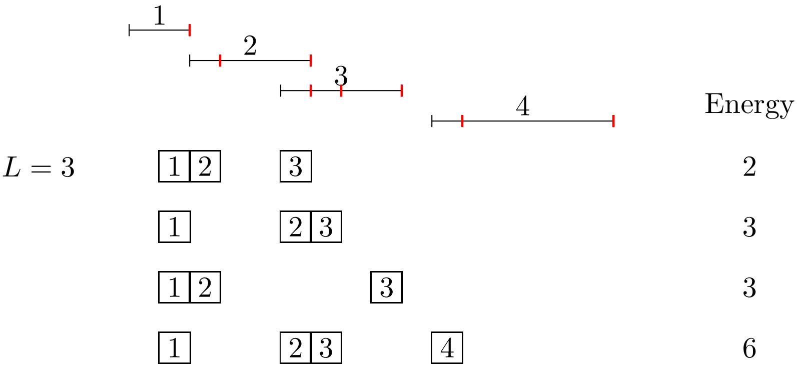
Example



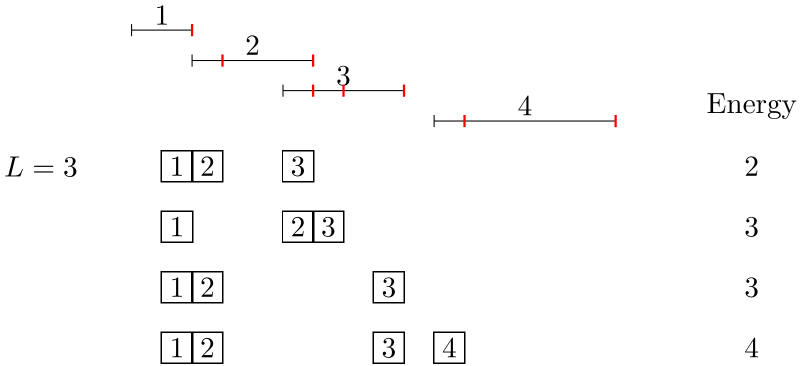
Example



Example



Example



Dynamic Programming

$E_1(d_1) = 0.$

$$\forall t \in D_k,\ E_k(t) = \begin{cases} \min\limits_{t' \in D_{k-1}} \{E_{k-1}(t') + \Delta(t', t - p_k)\} \text{ if } t = d_k \text{ or } t = r_k + p_k \\ E_{k-1}(t - p_k) \text{ Otherwise} \end{cases}$$

with

$$\Delta(t', t) = \begin{cases} \min\{L, t - t'\} \text{ si } t - t' \geq 0 \\ +\infty \text{ Otherwise} \end{cases}$$

Dynamic Programming

$E_1(d_1) = 0.$

$$\forall t \in D_k,\ E_k(t) = \begin{cases} \min_{t' \in D_{k-1}} \{E_{k-1}(t') + \Delta(t', t - p_k)\} \text{ if } t = d_k \text{ or } t = r_k + p_k \\ E_{k-1}(t - p_k) \text{ Otherwise} \end{cases}$$

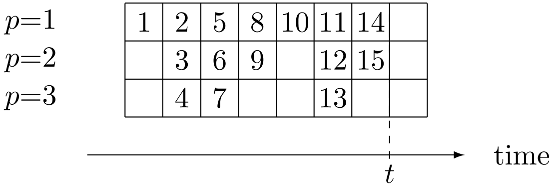
with

$$\Delta(t', t) = \begin{cases} \min\{L, t - t'\} \text{ si } t - t' \geq 0 \\ +\infty \text{ Otherwise} \end{cases}$$

Computed in $O(n^2)$

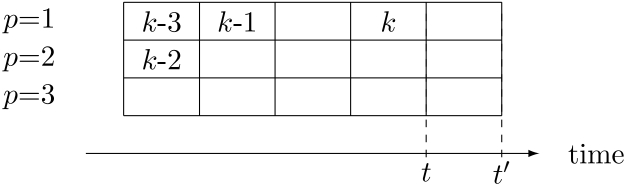
General Idea for Multiprocessor case

- Compact schedule
  - if a job j is scheduled at time t on a processor $p > 1$, then all lower-numbered processors $1 \leq q < p$ are also occupied at time t.
- EDF order
- Task k has additional positions : just before the task $k+1$
- Set D_k for task k still contains $O(n)$ positions.



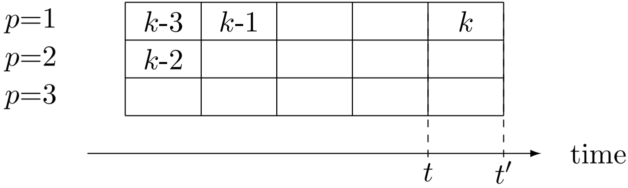
3 cases

Case 1 : $p = 1$



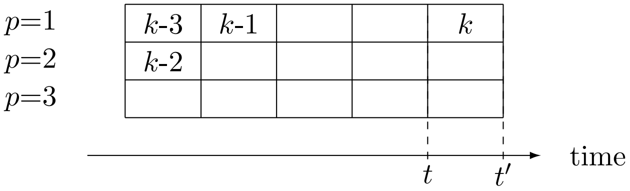
3 cases

Case 1 : $p = 1$

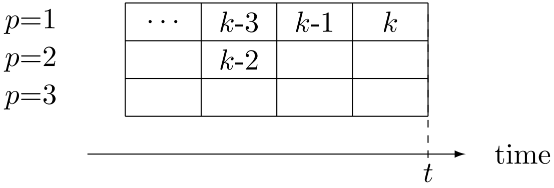


3 cases

Case 1 : $p = 1$

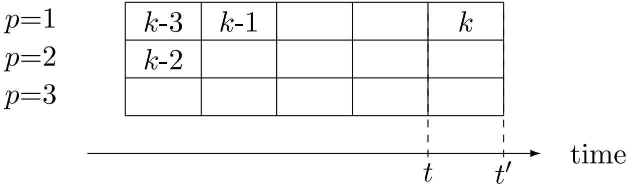


Case 2 : $p = 1$

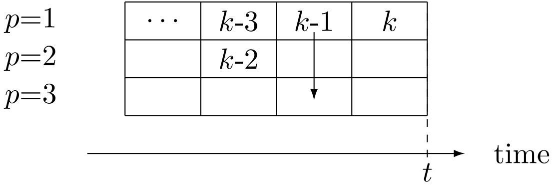


3 cases

Case 1 : $p = 1$

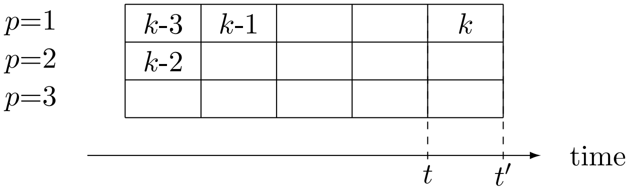


Case 2 : $p = 1$

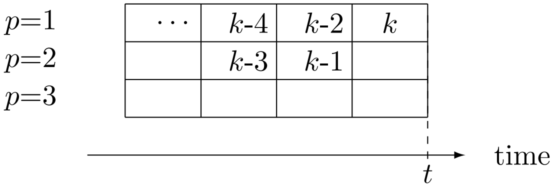


3 cases

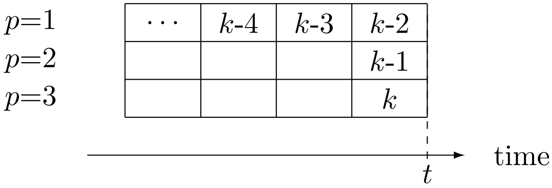
Case 1 : $p = 1$



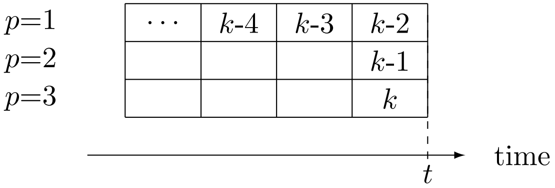
Case 2 : $p = 1$



Case 3 : $p > 1$



Case 3 : $p > 1$



Computed in $O(n^2m)$ time by Dynamic Programming

Outline

Open questions

- Multiprocessor case : can we have an algorithm for any L and for agreeable tasks ?
- Fast approximations : $1+\varepsilon$-approx. in $O(n)$ time ?

Thanks for your attention