Reclaiming the Energy of a Schedule, Models and Algorithms

Guillaume Aupy

based on a work done with Anne Benoit, Fanny Dufossé and Yves Robert

4th December 2012

Cours d’ordonnancement (CR-07)
1. Introduction
   Models
   Goal

2. Results
   Continuous speeds
   \texttt{Vdd-Hopping}
   Discrete speed models

3. Conclusion
• Scheduling = Makespan minimization
  Difficulty of scheduling is to chose the right processor to assign the task to.

• General mapping
  If we are not tight on deadline, why not take our time?
    • Economical + environmental reasons: Energy consumption.
    • Affinities or security reasons: what if the tasks are pre-assigned to a processor?

Goal: “efficiently” use speed scaling
Motivation

- **Scheduling = Makespan minimization**
  Difficulty of scheduling is to choose the right processor to assign the task to.

- **General mapping**
  If we are not tight on deadline, why not take our time?
  - Economical + environmental reasons: Energy consumption.
  - Affinities or security reasons: what if the tasks are pre-assigned to a processor?

Goal: “efficiently” use speed scaling
Motivation

• Scheduling – Makespan minimization
  Difficulty of scheduling is to choose the right processor to assign the task to.

• General mapping
  If we are not tight on deadline, why not take our time?
  • Economical + environmental reasons: Energy consumption.
  • Affinities or security reasons: what if the tasks are pre-assigned to a processor?

Goal: “efficiently” use speed scaling
Motivation

• Scheduling = Makespan minimization
  Difficulty of scheduling is to chose the right processor to assign the task to.

• General mapping
  If we are not tight on deadline, why not take our time?
  • Economical + environmental reasons: Energy consumption.
  • Affinities or security reasons: what if the tasks are pre-assigned to a processor?

Goal: “efficiently” use speed scaling
1 Introduction

   Models
   Goal

2 Results

   Continuous speeds
   VDD-HOPPING
   Discrete speed models

3 Conclusion
Consider a task graph (directed acyclic graph) to be executed on a set of processors. Assume that the mapping is given.

**Useful definition in a task graph**

For every task $T_i$ we define

- $w_i$: its size/work
- $s_i$: the speed of the processor which has task $T_i$ assigned to.
- $t_i$: the time when the computation of $T_i$ ends.
- $d_i$: the time it took to compute task $T_i$. 
- $d_i s_i^3$: the energy consumed on task $T_i$ by the system.
Consider a task graph (directed acyclic graph) to be executed on a set of processors. Assume that the mapping is given.

**Useful definition in a task graph**

For every task $T_i$ we define

- $w_i$: its size/work
- $s_i$: the speed of the processor which has task $T_i$ assigned to.
- $t_i$: the time when the computation of $T_i$ ends.
- $d_i$: the time it took to compute task $T_i$.
- $d_i s_i^3$: the energy consumed on task $T_i$ by the system.
Consider a task graph (directed acyclic graph) to be executed on a set of processors. Assume that the mapping is given.

**Useful definition in a task graph**

For every task $T_i$ we define

- $w_i$ its size/work
- $s_i$ the speed of the processor which has task $T_i$ assigned to.
- $t_i$ the time when the computation of $T_i$ ends.
- $d_i$ the time it took to compute task $T_i$.
- $d_i s_i^3$ the energy consumed on task $T_i$ by the system.
Consider a task graph (directed acyclic graph) to be executed on a set of processors. Assume that the mapping is given.

**Useful definition in a task graph**

For every task $T_i$ we define

- $w_i$: its size/work
- $s_i$: the speed of the processor which has task $T_i$ assigned to.
- $t_i$: the time when the computation of $T_i$ ends.
- $d_i$: the time it took to compute task $T_i$.
- $d_is_i^3$: the energy consumed on task $T_i$ by the system.
Consider a task graph (directed acyclic graph) to be executed on a set of processors. Assume that the mapping is given.

**Useful definition in a task graph**

For every task $T_i$ we define

- $w_i$: its size/work
- $s_i$: the speed of the processor which has task $T_i$ assigned to.
- $t_i$: the time when the computation of $T_i$ ends.
- $d_i$: the time it took to compute task $T_i$.
- $d_i s_i^3$: the energy consumed on task $T_i$ by the system.


- **Continuous**: any speeds in $[0, s_{max}]$. A processor can change speed at any time.

- **Discrete**: set of speed: $\{s_1, ..., s_m\}$. Constant speed during the computation of a task, but it can change from task to task.

- **Vdd-Hopping**: close to the previous model, difference: we can switch speeds during a computation.

- **Incremental**: Discrete model where $s_1 = s_{min}$, $s_m = s_{max}$, and for all $i$, $s_i = s_{min} + i \cdot \delta$ for some $\delta$. 
• **CONTINUOUS:** any speeds in \([0, s_{\text{max}}]\). A processor can change speed at any time.

**Gauss Fact**

*When Gauss wife asked him ”How much do you love me?”, he quantified it with an irrational number.*  
*Unfortunately a computer will never be as good as Gauss.*

• **DISCRETE:** set of speed: \(\{s_1, \ldots, s_m\}\). Constant speed during the computation of a task, but it can change from task to task.

• **Vdd-Hopping:** close to the previous model, difference: we can switch speeds during a computation.

• **INCREMENTAL:** Discrete model where \(s_1 = s_{\min}\), \(s_m = \frac{5.0}{s_{\max}}\), and for all \(i\), \(s_i = s_{\min} + i \cdot \delta\) for some \(\delta\).
• **CONTINUOUS:** any speeds in $[0, s_{max}]$. A processor can change speed at any time.

• **DISCRETE:** set of speed: $\{s_1, ..., s_m\}$. Constant speed during the computation of a task, but it can change from task to task.

• **VDD-HOPPING:** close to the previous model, difference: we can switch speeds during a computation.

• **INCREMENTAL:** DISCRETE model where $s_1 = s_{min}$, $s_m = s_{max}$, and for all $i$, $s_i = s_{min} + i \cdot \delta$ for some $\delta$. 
- **Continuous**: any speeds in $[0, s_{max}]$. A processor can change speed at any time.

- **Discrete**: set of speed: $\{s_1, ..., s_m\}$. Constant speed during the computation of a task, but it can change from task to task.

- **Vdd-Hopping**: close to the previous model, difference: we can switch speeds during a computation.

- **Incremental**: Discrete model where $s_1 = s_{min}$, $s_m = s_{max}$, and for all $i$, $s_i = s_{min} + i \cdot \delta$ for some $\delta$.  


• **CONTINUOUS**: any speeds in \([0, s_{\text{max}}]\). A processor can change speed at any time.

• **DISCRETE**: set of speed: \([s_1, ..., s_m]\). Constant speed during the computation of a task, but it can change from task to task.

• **Vdd-Hopping**: close to the previous model, difference: we can switch speeds during a computation.

• **Incremental**: **Discrete** model where \(s_1 = s_{\text{min}}\), \(s_m = s_{\text{max}}\), and for all \(i\), \(s_i = s_{\text{min}} + i \cdot \delta\) for some \(\delta\).
Consider this DAG, with $s_{max} = 6$. Suppose deadline is $D = 1.5$.

![Execution graph for the example.](image-url)
• **CONTINUOUS:** \((s_{\text{max}} = 6)~ E_{\text{opt}}^{(c)} \approx 109.6.\)

With the CONTINUOUS model, the optimal speeds are non-rational values, and we obtain

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

\[s_3 = s_4 = s_1 \times \frac{3}{35^{1/3}} \approx 3.83.\]

• **DISCRETE:** \((s_1 = 2, s_2 = 5, s_3 = 6)~ E_{\text{opt}}^{(d)} = 170.\)

• **INCREMENTAL:** \((\delta = 2, s_{\text{min}} = 2, s_{\text{max}} = 6)~ E_{\text{opt}}^{(i)} = 128.\)

• **VDD-HOPPING:** \((s_1 = 2, s_2 = 5, s_3 = 6)~ E_{\text{opt}}^{(v)} = 144.\)
• **Continuous**: \( s_{\text{max}} = 6 \) \( E^{(c)}_{\text{opt}} \approx 109.6 \).

• **Discrete**: \( (s_1 = 2, s_2 = 5, s_3 = 6) \) \( E^{(d)}_{\text{opt}} = 170 \).

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**: \( (\delta = 2, \ s_{\text{min}} = 2, \ s_{\text{max}} = 6) \) \( E^{(i)}_{\text{opt}} = 128 \).

• **Vdd-Hopping**: \( (s_1 = 2, s_2 = 5, s_3 = 6) \) \( E^{(v)}_{\text{opt}} = 144 \).
Example

- **Continuous**: \((s_{max} = 6)\) \(E^{(c)}_{opt} \approx 109.6\).
- **Discrete**: \((s_1 = 2, s_2 = 5, s_3 = 6)\) \(E^{(d)}_{opt} = 170\).
- **Incremental**: \((\delta = 2, s_{min} = 2, s_{max} = 6)\) \(E^{(i)}_{opt} = 128\). 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**: \((s_1 = 2, s_2 = 5, s_3 = 6)\) \(E^{(v)}_{opt} = 144\). 

- 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 \(s_2^{(i)} = 4\).
Example

- **Continuous**: \( s_{\text{max}} = 6 \) \( E_{opt}^{(c)} \approx 109.6 \).
- **Discrete**: \( s_1 = 2, \ s_2 = 5, \ s_3 = 6 \) \( E_{opt}^{(d)} = 170 \).
- **Incremental**: \( \delta = 2, \ s_{\text{min}} = 2, \ s_{\text{max}} = 6 \) \( E_{opt}^{(i)} = 128 \).
- **Vdd-Hopping**: \( s_1 = 2, \ s_2 = 5, \ s_3 = 6 \) \( E_{opt}^{(v)} = 144 \).

With the Vdd-Hopping model, we set \( s_1 = s_2^{(d)} = 5 \); for the other tasks, we run part of the time at speed \( s_2^{(d)} = 5 \), and part of the time at speed \( s_1^{(d)} = 2 \) in order to use the idle time and lower the energy consumption.
• **Continuous:** \((s_{\text{max}} = 6)\) \(E^{(c)}_{\text{opt}} \approx 109.6\).

• **Discrete:** \((s_1 = 2, s_2 = 5, s_3 = 6)\) \(E^{(d)}_{\text{opt}} = 170\).

• **Incremental:** \((\delta = 2, s_{\text{min}} = 2, s_{\text{max}} = 6)\) \(E^{(i)}_{\text{opt}} = 128\).

• **Vdd-Hopping:** \((s_1 = 2, s_2 = 5, s_3 = 6)\) \(E^{(v)}_{\text{opt}} = 144\).
Energy-Performance-oriented objective

- Constraint on Deadline
- Minimize Energy Consumption:

Today's talk: comparison of all speed models in this regard.

We assume the mapping is already fixed.
Energy-Performance-oriented objective

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

Today’s talk: comparison of all speed models in this regard.

We assume the mapping is already fixed.
Energy-Performance-oriented objective

- Constraint on Deadline \( t_i \leq D \) for each \( T_i \in V \)
- Minimize Energy Consumption: \( \sum_{i=1}^{n} w_i \times s_i^2 \)

Today’s talk: comparison of all speed models in this regard.

We assume the mapping is already fixed.
1 Introduction
   Models
   Goal

2 Results
   Continuous speeds
   Vdd-Hopping
   Discrete speed models

3 Conclusion
The problem of minimizing energy when the scheduled is already fixed on $p$ processors is:

- **Continuous**: Polynomial for some special graphs, geometric optimization in the general case.
- **Discrete**: NP-complete (reduction from 2-partition). We give an approximation.
- **Incremental**: NP-complete (reduction from 2-partition). We give an approximation.
- **Vdd-Hopping**: Polynomial (linear programming).
The problem of minimizing energy when the schedule is already fixed on $p$ processors is:

- **CONTINUOUS**: Polynomial for some special graphs, geometric optimization in the general case.
- **DISCRETE**: NP-complete (reduction from 2-partition). We give an approximation.
- **INCREMENTAL**: NP-complete (reduction from 2-partition). We give an approximation.
- **VDD-HOPPING**: Polynomial (linear programming).
The problem of minimizing energy when the scheduled is already fixed on $p$ processors is:

- **CONTINUOUS**: Polynomial for some special graphs, geometric optimization in the general case.
- **DISCRETE**: NP-complete (reduction from 2-partition). We give an approximation.
- **INCREMENTAL**: NP-complete (reduction from 2-partition). We give an approximation.
- **VDD-HOPPING**: Polynomial (linear programming).
The problem of minimizing energy when the scheduled is already fixed on $p$ processors is:

- **Continuous**: Polynomial for some special graphs, geometric optimization in the general case.
- **Discrete**: NP-complete (reduction from 2-partition). We give an approximation.
- **Incremental**: NP-complete (reduction from 2-partition). We give an approximation.
- **Vdd-Hopping**: Polynomial (linear programming).
The problem of minimizing energy when the scheduled is already fixed on $p$ processors is:

- **Continuous**: Polynomial for some special graphs, geometric optimization in the general case.
- **Discrete**: NP-complete (reduction from 2-partition). We give an approximation.
- **Incremental**: NP-complete (reduction from 2-partition). We give an approximation.
- **Vdd-Hopping**: Polynomial (linear programming).
Reminder
For each task $T_i$ we define

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

Objective function

Minimize $\sum_{i=1}^{n} s_i^2 \times w_i$

subject to (i) $t_i + \frac{w_j}{s_j} \leq t_j$ for each $(T_i, T_j) \in E$
(ii) $t_i \leq D$ for each $T_i \in V$
Results for continuous speeds

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

Definition

\[ G, \ n \text{ tasks}, \ D \text{ deadline}; \]
\[ s_1, \ldots, s_m \text{ be the set of possible processor speeds}; \]
\[ t_i \text{ is the finishing time of the execution of task } T_i; \]
\[ \alpha(i,j) \text{ is the time spent at speed } s_j \text{ for executing task } T_i \]
This makes us a total of \( n(m + 1) \) variables for the system.
Note that the total execution time of task \( T_i \) is \( \sum_{j=1}^{m} \alpha(i,j) \).
The objective function is:

\[
\min \left( \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha(i,j) s_j^3 \right)
\]
The constraints are:

∀1 ≤ i ≤ n, ti ≤ D: the deadline is not exceeded by any task;

∀1 ≤ i, i′ ≤ n s.t. Ti → Ti′, ti + \sum_{j=1}^{m} \alpha(i′, j) ≤ ti′: a task cannot start before its predecessor has completed its execution;

∀1 ≤ i ≤ n, \sum_{j=1}^{m} \alpha(i, j) × sj ≥ wi: task Ti is completely executed.

∀1 ≤ i ≤ n, ti ≥ \sum_{j=1}^{m} \alpha(i, j): each task cannot finish until all work is done;
Theorem

With the **Incremental model** (and hence the **Discrete model**), finding the speed distribution that minimizes the energy consumption while enforcing a deadline $D$ is NP-complete.
Theorem

With the Incremental model (and hence the Discrete model), finding the speed distribution that minimizes the energy consumption while enforcing a deadline \( D \) is NP-complete.

PROOF: Reduction from 2-Partition,

- 1 processor, \( n \) independent tasks of weight \( (a_i) \).
- 2 speeds: \( s_1 = 1/2, s_2 = 3/2 \)
- \( D = 2W = \sum_{i=1}^{n} a_i \)
- \( E = W((3/2)^2 + (1/2)^2) \)
Approximation results for **Discrete** and **Incremental**.

Proposition (Polynomial-time Approximation algorithms.)

- **With the Discrete model**, for any integer $K > 0$, the $\text{MinEnergy}(G,D)$ problem can be approximated within a factor

  $$(1 + \frac{\alpha}{s_1})^2 \times (1 + \frac{1}{K})^2$$

  where $\alpha = \max_{1 \leq i < m} \{s_{i+1} - s_i\}$, in a time polynomial in the size of the instance and in $K$.

- **With the Incremental model**, the same result holds where $\alpha = \delta \ (s_1 = s_{\text{min}})$. 
Approximation results for Discrete and Incremental.

Proposition (Comparison to the optimal solution:)

For any integer $\delta > 0$, any instance of $\text{MinEnergy}(G,D)$ with the Continuous model can be approximated within a factor $(1 + \frac{\delta}{s_{\text{min}}})^2$ in the Incremental model with speed increment $\delta$. 
Introduction

Models
Goal

Results
Continuous speeds
Vdd-Hopping
Discrete speed models

Conclusion
The problem of minimizing energy when the scheduled is already fixed on \( p \) processors is:

**CONTINUOUS**: Polynomial for some special graphs, geometric optimization in the general case.

**DISCRETE and INCREMENTAL**: NP-complete. However we were able to give an approximation.

**VDD-HOPPING**: Polynomial (linear programming).

- Bi-criteria Energy/Deadline optimization problem
- Mapping already given.
- Theoretical foundations for a comparative study of energy models.
Thanks for listening. Any questions?