Energy-aware algorithms

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr
http://graal.ens-lyon.fr/~abenoit

CR02 - 2016/2017
**Speed models for DVFS**

<table>
<thead>
<tr>
<th>Type of speeds</th>
<th>When can we change speed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[s_{\text{min}}, s_{\text{max}}]$</td>
<td>CONTINUOUS</td>
</tr>
<tr>
<td>${s_1, \ldots, s_m}$</td>
<td>VDD-HOPPING</td>
</tr>
<tr>
<td></td>
<td>BEGINNING OF TASKS</td>
</tr>
</tbody>
</table>

- **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 $1 \leq q < m$, $s_{q+1} - s_q = \delta$
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

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

- \textbf{MinEnergy}(G,D) can be solved in polynomial time when \( G \) is a tree.

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

\textbf{TODO:} Prove the lemma for forks and joins to prove that \textbf{MinEnergy}(G,D) can be solved in polynomial time in this case (we just need to find \( s_0 \)).
Linear program for **VDD-HOPPING**

**Definition**

- $G$, $n$ tasks, $D$ deadline;
- $s_1, \ldots, s_m$ be the set of possible processor speeds;
- $t_i$ is the finishing time of the execution of task $T_i$;
- $\alpha(i,j)$ is the *time* spent at speed $s_j$ 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:

$$
\min \left( \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha(i,j) s_j^3 \right)
$$
Linear program for \textbf{VDD-Hopping}

The constraints are:

\( \forall 1 \leq i \leq n, \ t_i \leq D \): the deadline is not exceeded by any task;

\( \forall 1 \leq i, i' \leq n \text{ s.t. } T_i \rightarrow T_{i'}, \ t_i + \sum_{j=1}^{m} \alpha(i',j) \leq t_{i'} \): a task cannot start before its predecessor has completed its execution;

\( \forall 1 \leq i \leq n, \ \sum_{j=1}^{m} \alpha(i,j) \times s_j \geq w_i \): task \( T_i \) is completely executed;

\( \forall 1 \leq i \leq n, \ t_i \geq \sum_{j=1}^{m} \alpha(i,j) \): each task cannot finish until all work is done.
NP-completeness for discrete speed models

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: $s_1 = 1$, $s_2 = 2$ (increment of 1)
- $D = 3T/2$ (where $T = \frac{1}{2} \sum_{i=1}^{n} a_i$)
- $E = 5T$
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 (Comparaison 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 $\delta$. 
Summary

- Results for **Continuous**, but not very practical

- In real life, **Discrete** model (DVFS)

- **Vdd-Hopping**: good alternative, mixing two consecutive modes, smoothes out the discrete nature of modes

- **Incremental**: alternate (and simpler in practice) solution, with one unique speed during task execution; can be made arbitrarily efficient
Outline

1. Introduction and motivation: energy
2. Revisiting the greedy algorithm for independent jobs
3. Reclaiming the slack of a schedule
4. Conclusion
What we had:

Energy-efficient scheduling + frequency scaling

What we aim at:
Thanks...

...to my co-authors Guillaume Aupy, Fanny Dufossé, Paul Renaud-Goud, and Yves Robert.

Bibliography:

- On the performance of greedy algorithms for energy minimization (Benoit, Renaud-Goud, Robert, 2011)
- Reclaiming the energy of a schedule: models and algorithms (Aupy, Benoit, Dufossé, Robert, 2013)
Performance and energy optimization of concurrent pipelined applications

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr
http://graal.ens-lyon.fr/~abenoit

CR02 - 2016/2017
Motivations

- Mapping of concurrent pipelined applications on parallel platform: practical applications, but difficult problem

  → classification of mappings and platforms

- Energy saving is becoming a crucial problem

- Objective functions: period, latency, power

- Multi-criteria approach

- Complexity study
Outline of the talk

1. Definitions
2. Mono-criterion problems
3. Bi-criteria problems
4. Tri-criteria problems
5. Conclusion
Outline

1. Definitions
2. Mono-criterion problems
3. Bi-criteria problems
4. Tri-criteria problems
5. Conclusion
Motivating example

- Period: $T = 3$
- Latency: $L = 8$
Motivating example

- Period: $T = 3$
- Latency: $L = 8$
Motivating example

- Period: \( T = 3 \)
- Latency: \( L = 8 \)

\[ P = 3^3 + 8^3 = 539 \]
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

**Period:** \( T = 3 \)

**Latency:** \( L = 8 \)
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

- Period: \( T = 3 \)
- Latency: \( L = 8 \)
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

- Period: \( T = 3 \)
- Latency: \( L = 8 \)
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

Period: \( T = 3 \)
Latency: \( L = 8 \)
Motivating example

Period: $T = 3$
Latency: $L = 8$

$P = 3^3 + 8^3$
$= 539$
Motivating example

$P = 3^3 + 8^3 = 539$

- Period: $T = 3$
- Latency: $L = 8$
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

- Period: \( T = 3 \)
- Latency: \( L = 8 \)
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

- Period: \( T = 3 \)
- Latency: \( L = 8 \)
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

- Period: \( T = 3 \)
- Latency: \( L = 8 \)
Motivating example

Period: $T = 3$

Latency: $L = 8$

$$P = 3^3 + 8^3 = 539$$
Motivating example

- Period: \( T = 3 \)
- Latency: \( L = 8 \)

\[ P = 3^3 + 8^3 = 539 \]
Motivating example

- Period: $T = 3$
- Latency: $L = 8$

$$P = 3^3 + 8^3 = 539$$
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

- Period: \( T = 3 \)
- Latency: \( L = 8 \)
Motivating example

- Period: $T = 3$
- Latency: $L = 8$

$$P = 3^3 + 8^3 = 539$$
Motivating example

\[ P = 3^3 + 8^3 = 539 \]

- Period: \( T = 3 \)
- Latency: \( L = 8 \)
Motivating example

Period: $T = 3$
Latency: $L = 8$
Motivating example

Period: $T = 3 \quad T = 15$

Latency: $L = 8$
Motivating example

Period: $T = 3 \quad T = 15$
Latency: $L = 8 \quad L = 17$
Applications and platform

- For an application $a$:
  - $w^i_a$: weight of stage $S^i_a$
  - $\delta^i_a$: size of outcoming data of $S^i_a$

- Processors with multiple speeds (or modes): \(\{s_{u,1}, \ldots, s_{u,m_u}\}\)
  - Constant speed during the execution
  - $b_{u,v}$: bandwidth between processors $P_u$ and $P_v$

- Platform fully interconnected

- Communications: both overlap or non-overlap model

- Three platforms types:
  1. Fully homogeneous
  2. Communication homogeneous
  3. Fully heterogeneous
Mappings

No processor sharing for both practical and theoretical reasons (security rules and NP-completeness of the execution scheduling given a mapping with a period/latency objective).

- One-to-one mapping

- Interval mapping
No processor sharing for both practical and theoretical reasons (security rules and NP-completeness of the execution scheduling given a mapping with a period/latency objective).

- **One-to-one mapping**

- **Interval mapping**
No processor sharing for both practical and theoretical reasons (security rules and NP-completeness of the execution scheduling given a mapping with a period/latency objective).

- **One-to-one mapping**

- **Interval mapping**

  $\text{App}_1$  
  $\text{App}_2$
Interval mapping on a single application; \(k\) intervals \(l_j\) of stages from \(S^{d_j}\) to \(S^{e_j}\); \(\alpha\) assignment procedure

- **Period** \(T\) of an application: the minimum delay between the processing of two consecutive set of data

\[
T^{(\text{overlap})} = \max_{j \in \{1, \ldots, k\}} \left( \max \left( \frac{\delta^{d_j} - 1}{b_{\alpha(d_j - 1), \alpha(d_j)}}, \frac{\sum_{i=d_j}^{e_j} w^i}{s_{\alpha(d_j)}}, \frac{\delta^{e_j}}{b_{\alpha(d_j), \alpha(e_j + 1)}} \right) \right)
\]

- **Latency** \(L\) of an application: time, for a data set, to go through the whole pipeline

\[
L = \frac{\delta^0}{b_{\alpha(0), \alpha(1)}} + \sum_{j=1}^{m} \left( \sum_{i=d_j}^{e_j} \frac{w^i}{s_{\alpha(d_j)}} + \frac{\delta^{e_j}}{b_{\alpha(d_j), \alpha(e_j + 1)}} \right)
\]

- **Power of a processor** \(P_u\):

\[
P(u) = P_{\text{dyn}}(s_u) + P_{\text{stat}}(u) \quad ; \quad P_{\text{dyn}}(s_u) = s_u^\alpha
\]
Optimization problems

- Minimize one criterion:
  - Period or latency: minimize $\max_a W_a \times T_a$ or $\max_a W_a \times L_a$
  - Power: minimize $\sum_u P(u)$

- Fix one criterion:
  - Fix the period or latency of each application $\rightarrow$ fix a period or latency array
  - Fix $\sum_u P(u)$

- Multi-criteria approach: minimizing 1 criterion, fixing the other ones

- Power consumption, i.e., energy per time unit $\Rightarrow$ combination power/period
Outline

1. Definitions
2. Mono-criterion problems
3. Bi-criteria problems
4. Tri-criteria problems
5. Conclusion
## Complexity results

### Period minimization:

<table>
<thead>
<tr>
<th></th>
<th>proc-hom</th>
<th>special-app(^1)</th>
<th>proc-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>com-hom</td>
<td>polynomial (binary search)</td>
<td>NP-complete</td>
<td></td>
</tr>
<tr>
<td>one-to-one</td>
<td>polynomial</td>
<td>NP-complete</td>
<td></td>
</tr>
<tr>
<td>interval</td>
<td>polynomial</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

\(^1\)special-app: com-hom & pipe-hom
Period minimization - com-hom - one-to-one

Problem: one-to-one mapping - many applications - communication homogeneous platform - period minimization

Algorithm 1: Greedy-Assignment(T)

begin
    Work with fastest $N$ processors, numbered $P_1$ to $P_N$, where $s_1 \leq s_2 \leq \cdots \leq s_N$;
    Mark all stages as free;
    for $u = 1$ to $N$ do
        Pick up any free stage $S_a^k$ s.t. $W_a \times \max (\frac{\delta_a^{k-1}}{b}, \frac{w_a^k}{s_u}, \frac{\delta_a^k}{b}) \leq T$;
        Assign $S_a^k$ to $P_u$;
        Mark $S_a^k$ as already assigned;
        if no stage found then
            return "failure";
        end
    end
    return "success";
end
Period minimization - interval

- Polynomial for fully homogeneous platforms, building upon optimal algorithm for a single application
- NP-complete even with a homogeneous application with heterogeneous processors
NP-complete! Involved reduction from MINIMUM METRIC BOTTLENECK WANDERING SALESPERSON PROBLEM:

- Set of \( m \) cities \( c_1, \ldots, c_m \)
- Distances \( d(c_i, c_j) \) satisfying the triangle inequality
- Find a simple path from \( c_1 \) to \( c_m \), while minimizing the maximum distance in the path
### Complexity results

#### Period minimization:

<table>
<thead>
<tr>
<th></th>
<th>proc-hom</th>
<th>com-hom</th>
<th>special-app&lt;sup&gt;1&lt;/sup&gt;</th>
<th>proc-het</th>
<th>com-hom</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-to-one</td>
<td></td>
<td></td>
<td>polynomial (binary search)</td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
<tr>
<td>interval</td>
<td>polynomial</td>
<td></td>
<td>NP-complete</td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

#### Latency minimization:

<table>
<thead>
<tr>
<th></th>
<th>proc-hom</th>
<th>com-hom</th>
<th>special-app&lt;sup&gt;1&lt;/sup&gt;</th>
<th>proc-het</th>
<th>com-hom</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-to-one</td>
<td>polynomial</td>
<td></td>
<td>NP-complete</td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
<tr>
<td>interval</td>
<td>polynomial (binary search)</td>
<td></td>
<td>NP-complete</td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

<sup>1</sup>special-app: com-hom & pipe-hom
Latency minimization

- Problem: one-to-one mapping - many applications - heterogeneous platform - no communication - homogeneous pipelines - minimize $\max_a L_a$

- Single application: greedy polynomial algorithm

- Many applications: reduction from 3-PARTITION

3-PARTITION:
- Input: $3m + 1$ integers $a_1, a_2, \ldots, a_{3m}$ and $B$ such that $\sum_i a_i = mB$

- Does there exist a partition $l_1, \ldots, l_m$ of $\{1, \ldots, 3m\}$ such that for all $j \in \{1, \ldots, m\}$, $|l_j| = 3$ and $\sum_{i \in l_j} a_i = B$?
Latency minimization (2)

- **3-PARTITION**: does there exist a renumbering of \( a_i \) such that:

\[
\begin{align*}
    a_{1,1} + a_{1,2} + a_{1,3} &= B \\
    a_{2,1} + a_{2,2} + a_{2,3} &= B \\
    &\vdots \\
    a_{m,1} + a_{m,2} + a_{m,3} &= B
\end{align*}
\]

- **Reduction**:

Can we obtain a latency \( L^0 \leq B \)?

- **Equivalence of problems**
Outline

1. Definitions
2. Mono-criterion problems
3. Bi-criteria problems
4. Tri-criteria problems
5. Conclusion
### Complexity results

#### Period/latency minimization:

<table>
<thead>
<tr>
<th></th>
<th>proc-hom</th>
<th>special-app</th>
<th>proc-het</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-to-one</td>
<td>polynomial</td>
<td></td>
<td>NP-complete</td>
<td></td>
</tr>
<tr>
<td>or interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Power/period minimization:

<table>
<thead>
<tr>
<th></th>
<th>proc-hom</th>
<th>special-app</th>
<th>proc-het</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-to-one</td>
<td>polynomial (minimum matching)</td>
<td>NP-complete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interval</td>
<td>polynomial</td>
<td></td>
<td>NP-complete</td>
<td></td>
</tr>
</tbody>
</table>
### Complexity results

#### Period/latency minimization:

<table>
<thead>
<tr>
<th>One-to-one or interval</th>
<th>proc-hom</th>
<th>com-hom</th>
<th>special-app</th>
<th>proc-het</th>
<th>com-hom</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-to-one or interval</td>
<td>polynomial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

#### Power/period minimization:

<table>
<thead>
<tr>
<th>One-to-one or interval</th>
<th>proc-hom</th>
<th>com-hom</th>
<th>special-app</th>
<th>proc-het</th>
<th>com-hom</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-to-one or interval</td>
<td>polynomial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
<tr>
<td>interval</td>
<td>polynomial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
</tbody>
</table>
Power/period minimization

- Problem: one-to-one mapping - many applications - communication homogeneous platform - power minimization for a given array of periods
- Minimum weighted matching of a bipartite graph

\[ p \geq N \]
### Complexity results

#### Period/latency minimization:

<table>
<thead>
<tr>
<th></th>
<th>proc-hom</th>
<th>com-hom</th>
<th>special-app</th>
<th>proc-het</th>
<th>com-hom</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-to-one</td>
<td></td>
<td></td>
<td>polynomial</td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
<tr>
<td>or interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Power/period minimization:

<table>
<thead>
<tr>
<th></th>
<th>proc-hom</th>
<th>com-hom</th>
<th>special-app</th>
<th>proc-het</th>
<th>com-hom</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>one-to-one</td>
<td></td>
<td></td>
<td>polynomial (minimum matching)</td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
<tr>
<td>interval</td>
<td></td>
<td></td>
<td>polynomial</td>
<td></td>
<td></td>
<td>NP-complete</td>
</tr>
</tbody>
</table>
Single application

- Problem: interval mapping - single application - fully homogeneous platform - power minimization for a given period

- \( P(i, j, k) \): minimum power to run stages \( S^i \) to \( S^j \) using exactly \( k \) processors \( \rightarrow \) looking for \( \min_{1 \leq k \leq p} P(1, n, k) \)

- Recurrence relation:

\[
P(i, j, k) = \min_{1 \leq \ell \leq j-1} (P(i, \ell, k - 1) + P(\ell + 1, j, 1))
\]
Single application (2)

- $P(i, i, q) = +\infty$ if $q > 1$

- $F_i^j$: possible powers of a processor running the stages $S_i^j$ to $S_j^j$, fulfilling the period constraint

- $F_i^j = \left\{ P_{dyn}(s_\ell) + P_{stat}, \max \left( \frac{\delta^{i-1}}{b}, \sum_{k=i}^{j} \frac{w_k}{s_\ell}, \frac{\delta_j}{b} \right) \right\} \leq T, \ell \in \{1, \ldots, m\}$

- $P(i, j, 1) = \begin{cases} \min F_i^j & \text{if } F_i^j \neq \emptyset \\ +\infty & \text{otherwise} \end{cases}$
Many applications

- Problem: interval mapping - fully homogeneous platform - power minimization for given periods by application

- $P^q_a$: minimum power consumed by $q$ processors so that the period constraint on the application $a$ is met, found by the previous dynamic programming

- $P(a, k)$: minimum power consumed by $k$ processors on the applications $1, \ldots, a$, unknown

- Initialization: $\forall k \in \{1, \ldots, p\}$ \hspace{0.5cm} $P(1, k) = P^k_1$
Many applications (2)

- Recurrence: \( P(a, k) = \min_{1 \leq q < k} (P(a - 1, k - q) + P^q_a) \)
Outline

1. Definitions
2. Mono-criterion problems
3. Bi-criteria problems
4. Tri-criteria problems
5. Conclusion
## Complexity results

<table>
<thead>
<tr>
<th>proc-hom</th>
<th>com-hom</th>
<th>proc-het</th>
<th>com-het</th>
</tr>
</thead>
<tbody>
<tr>
<td>special-app</td>
<td>com-hom</td>
<td>com-het</td>
<td></td>
</tr>
<tr>
<td>one-to-one or interval</td>
<td></td>
<td>NP-complete</td>
<td></td>
</tr>
</tbody>
</table>

Reduction from **2-PARTITION**

(Instance of **2-PARTITION**: $a_1, a_2, \ldots, a_n$ with $\sigma = \sum_{i=1}^{n} a_i$)
Problem instance

One-to-one mapping - fully homogeneous platform

\[
\begin{align*}
S_{2n} & = K^i \\
S_{2n-1} & = K^i + \frac{a_i}{K^{i(\alpha-1)}} X \\
S_{6} & = K^i \\
S_{5} & = \ldots \\
S_{4} & = \ldots \\
S_{3} & = \ldots \\
S_{2} & = \ldots \\
S_{1} & = \ldots \\
\end{align*}
\]

\[w_i = K^{i(\alpha+1)}\]

\[P^0 = P^* + \alpha X (\sigma/2 + 1/2), \quad L^0 = L^* - X (\sigma/2 - 1/2), \quad T^0 = L^0\]

where \(P^*\) and \(L^*\) are power and latency when each \(S_i\) is run at speed \(s_{2i-1}\)
Main ideas

- K big enough and X small enough so that the stage $S_i$ must be processed at speed $s_{2i-1}$ or $s_{2i}$

- For a subset $\mathcal{I}$ of $\{1, \ldots, n\}$, if ($S_i$ is run at speed $s_{2i} \iff i \in \mathcal{I}$),

$$P = P^* + \sum_{i \in \mathcal{I}} (\alpha a_i X + o(X)) \quad , \quad L = L^* - \sum_{i \in \mathcal{I}} (a_i X - o(X))$$

- Recall:

$$P^0 = P^* + \alpha X(\sigma/2 + 1/2) \quad , \quad L^0 = L^* - X(\sigma/2 - 1/2)$$
Outline

1. Definitions
2. Mono-criterion problems
3. Bi-criteria problems
4. Tri-criteria problems
5. Conclusion
Conclusion

- New polynomial algorithms for a single application
- Polynomial algorithms for a single application extended to many applications
- New results of NP-completeness
- Exhaustive complexity study

- Bibliography: Models and complexity results for performance and energy optimization of concurrent streaming applications (Benoit, Renaud-Goud, Robert, 2011)