Minimize the cardinality of a real-time task set through Task Clustering

Antoine Bertout, Julien Forget and Richard Olejnik

Laboratoire d'Informatique Fondamentale de Lille (LIFL)
Univ. Lille, France
RTNS'2014 Versailles

Outline

Outline

Context

- Focus on hard real-time systems
- Interest in programming of large systems specified as high-level functionalities

- Up to $\approx$ 1000 **high-level functionalities** in RT system software (e.g. aileron command, read pressure sensor, etc.)

Task Clustering

Problem

- Functionalities implemented via real-time threads (tasks) by programmers
- RT operating systems (OS) support a **limited number** of concurrent threads (several tens of OS tasks)

Task Clustering

RTOS limitations

Having numerous threads:

- Scheduling overhead
  - Scheduler level: handle large queues
  - Increase of context switches
    $\Rightarrow$ increase the risk of cache misses (larger WCET)
- Memory
  - Task level: one stack by task
  - Scheduler level: increase in the number of priorities required
    $\Rightarrow$ number of preemption increases
    $\Rightarrow$ execution stack grows

Task Clustering

RTOS limitations

Having numerous threads:

- Scheduling overhead
  - Scheduler level: handle large queues
  - Increase of context switches
    $\Rightarrow$ increase the risk of cache misses (larger WCET)
- Memory
  - Task level: one stack by task
  - Scheduler level: increase in the number of priorities required
    $\Rightarrow$ number of preemption increases
    $\Rightarrow$ execution stack grows

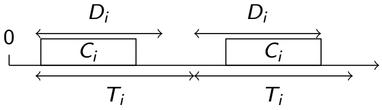
$\rightarrow$ **Several functionalities grouped together in a thread**

Manually made in industry (error prone, tedious)

Outline

System model used

Program = a set of tasks τ_i :

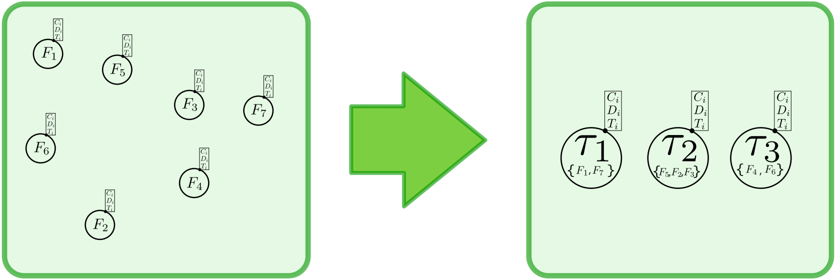


- T_i: period
- D_i: constrained relative deadline ($D_i \leq T_i$)
- C_i: worst case execution time (WCET)

Current limitations: independent and synchronous tasks (offset = 0) in a uniprocessor system.
Fixed task-priority and fixed-job-priority assignment considered.

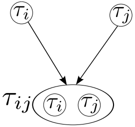
Objective

- Automatically grouping functionalities into tasks to minimize their number:
- while respecting original timing constraints,
- while **preserving schedulability**.



Cluster model

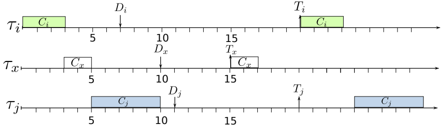
- **Cluster** τ_i and τ_j into τ_{ij} with $D_i \leq D_j$



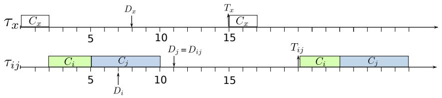
  - $C_{ij} = C_i + C_j$
  - $T_{ij} = T_i = T_j$ (by restriction we only regroup tasks with equal periods)
  - Which deadline for the cluster?

Cluster model : Deadline choice for the cluster

- Case 1: Maximum deadline D_j
  - if $(D_j - C_j \leq D_i)$
  - and τ_{ij} $\boxed{\tau_i \mid \tau_j}$ in that order



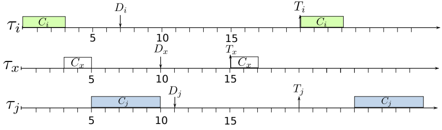
(a) Initial system with tasks τ_i,τ_x et τ_j



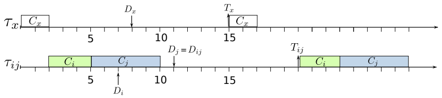
(b) System after clustering τ_i with τ_j

Cluster model : Deadline choice for the cluster

- Case 1: Maximum deadline D_j
  - if $(D_j - C_j \leq D_i)$
  - and τ_{ij} $\boxed{\tau_i \mid \tau_j}$ in that order



(a) Initial system with tasks τ_i,τ_x et τ_j

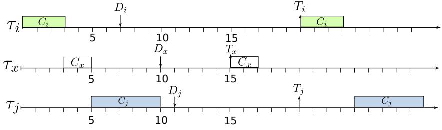


(b) System after clustering τ_i with τ_j

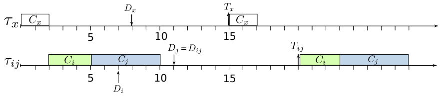
Schedulability preserved $\Rightarrow$ **Zero-cost clustering**

Cluster model : Deadline choice for the cluster

- Case 1: Maximum deadline D_j
  - if $(D_j - C_j \leq D_i)$ *or generalizing* $(R_j - C_j \leq D_i)$
  - and τ_{ij} $\boxed{\tau_i \mid \tau_j}$ in that order



(a) Initial system with tasks τ_i,τ_x et τ_j

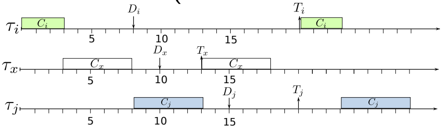


(b) System after clustering τ_i with τ_j

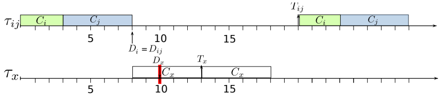
Schedulability preserved $\Rightarrow$ **Zero-cost clustering**

Cluster model : Deadline choice for the cluster

- Case 2: Minimum deadline D_i
  - Taking minimum deadline ensures respect of both initial ones
  - τ_{ij} $\boxed{\tau_i \mid \tau_j}$ or τ_{ij} $\boxed{\tau_j \mid \tau_i}$ (order does not matter)



(a) Initial system with tasks τ_i,τ_x et τ_j

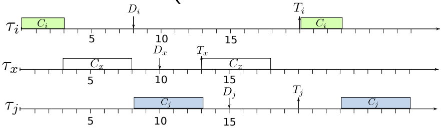


(b) System after clustering τ_i with τ_j

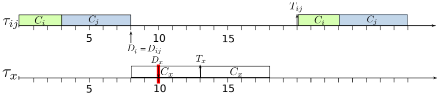
- System may become unschedulable after clustering

Cluster model : Deadline choice for the cluster

- Case 2: Minimum deadline D_i
  - Taking minimum deadline ensures respect of both initial ones
  - τ_{ij} $\boxed{\tau_i \mid \tau_j}$ or τ_{ij} $\boxed{\tau_j \mid \tau_i}$ (order does not matter)



(a) Initial system with tasks τ_i,τ_x et τ_j



(b) System after clustering τ_i with τ_j

- System may become unschedulable after clustering

$\Rightarrow$ Schedulability must be checked after each non zero-cost clustering!

Cluster model

Valid cluster

Theorem

Let $\mathcal{S}$ be a task set and Φ be a priority assignment. Let $\mathcal{S}'$ the task set $\mathcal{S}$ after clustering of two tasks τ_i and τ_j.
$\mathcal{S}'$ **is schedulable under** $\Phi \Rightarrow \mathcal{S}$ **is schedulable under** Φ**.**

However, the converse is not always true.

Definition

The clustering is valid iff schedulability is preserved after clustering

Outline

Complexity: a schedulability problem

Reminder:

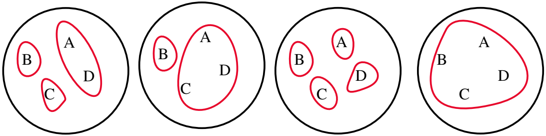
- **sufficient** test
  - often in linear complexity
  - ensures system schedulability
  - sub-optimal
- **necessary** test
  - does not ensure schedulability
- **exact** $\Rightarrow$ sufficient AND necessary test
  - generally **NP-hard**

In our case, use of sufficient or exact tests to guarantee system correctness

Complexity: and a partitionning problem

Example

Some possible combinations (15 possibilities) of 4 tasks τ_a, τ_b, τ_c et τ_d :



- **Combinatorial explosion**: number of possible clusterings in the Bell number range (e.g.,$\mathbf{B_{500} = 10^{844}}$)

Overall Complexity

- **NP-hard**: reduced from bin-packing with fragile objects
- Given:
  - a set of bins of **capacity** c_i
  - a set of object with a **size** s_i and a **fragility** f_i
- Assign objects to a minimum number of bins such that for each bin j:
  - $\sum s_i \leq c_j$
  - $\sum s_i \leq \min(f_i)$
- Task clustering analogy:
  - bin $\Rightarrow$ cluster
  - object $\Rightarrow$ task
  - fragility $\Rightarrow$ deadline

Overall Complexity

- **NP-hard**: reduced from bin-packing with fragile objects
- Given:
  - a set of bins of **capacity** c_i
  - a set of object with a **size** s_i and a **fragility** f_i
- Assign objects to a minimum number of bins such that for each bin j:
  - $\sum s_i \leq c_j$
  - $\sum s_i \leq \min(f_i)$
- Task clustering analogy:
  - bin $\Rightarrow$ cluster
  - object $\Rightarrow$ task
  - fragility $\Rightarrow$ deadline

$\rightarrow$ Motivates to work towards a **heuristic**

Outline

Heuristic principles

- Classical optimization techniques are based on cost functions to choose "promising candidates"
  - $\Rightarrow$ Schedulability test used as a cost function.
    (Applied to greedy BFS, but works for simulated annealing, A*, etc.)
- Perform in priority zero-cost clustering
- Use sufficient test when exact impracticable

A greedy heuristic

- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).

- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).

- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

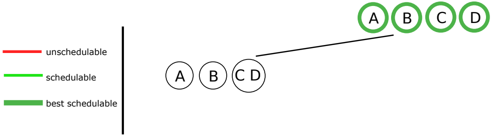
A greedy heuristic

- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$
  ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).

- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

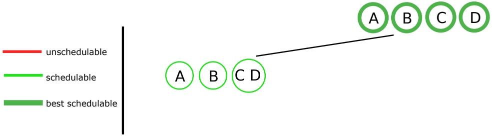
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

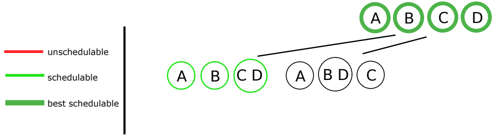
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

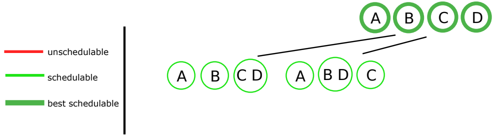
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

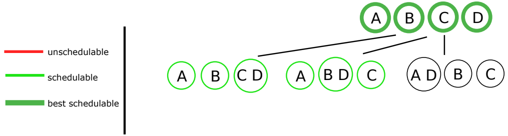
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

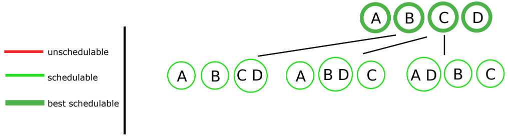
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

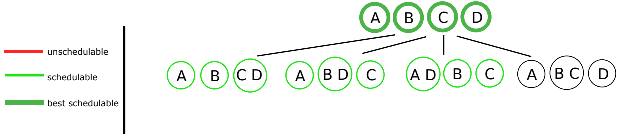
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

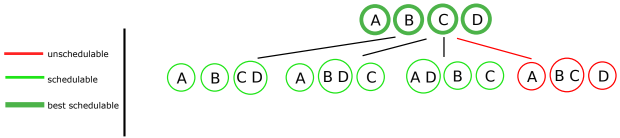
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

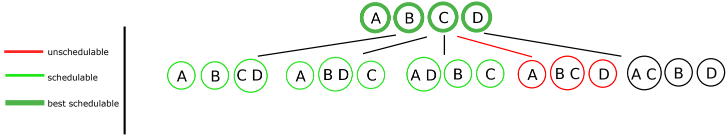
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering ⇒ avoid useless search

A greedy heuristic

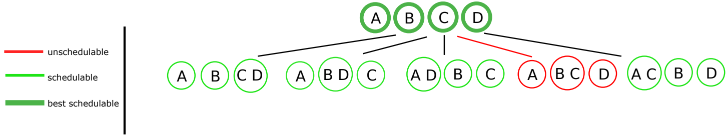
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering ⇒ avoid useless search

A greedy heuristic

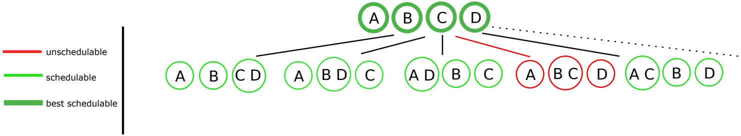
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

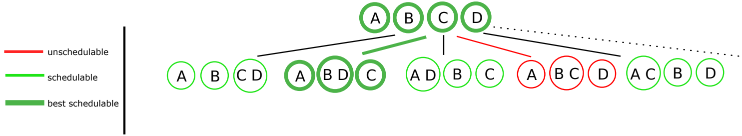
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

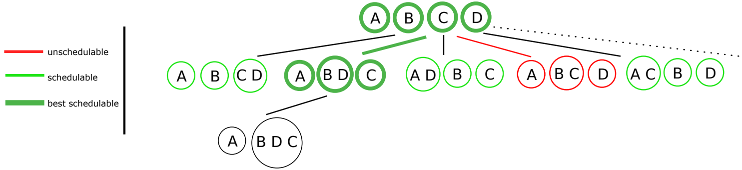
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

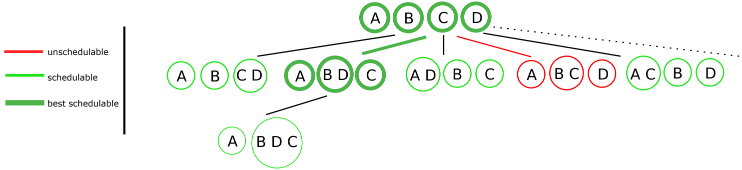
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

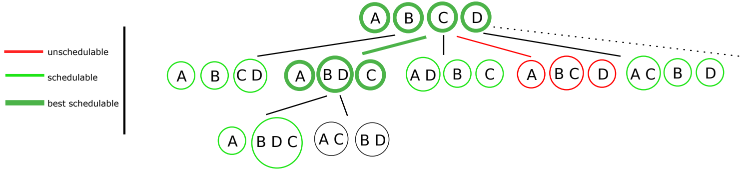
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

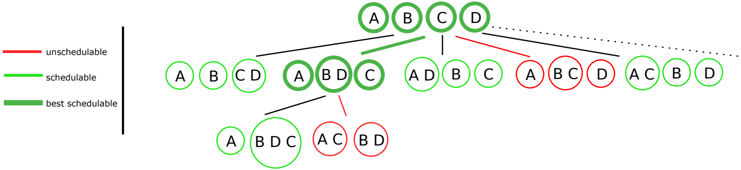
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

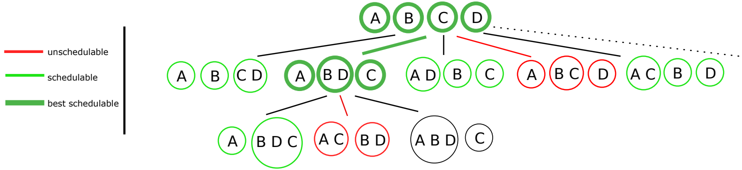
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering ⇒ avoid useless search

A greedy heuristic

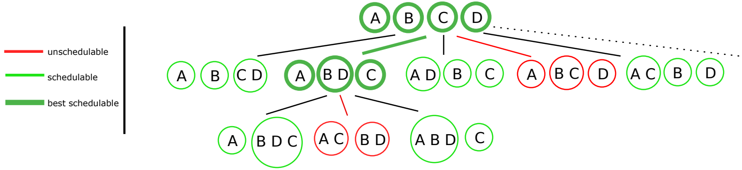
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$
  ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

A greedy heuristic

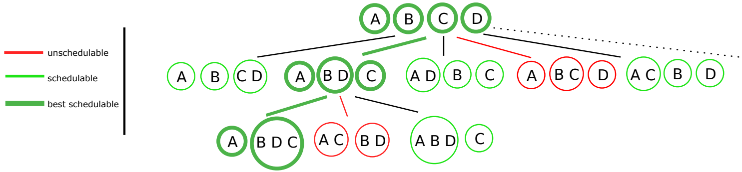
- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).



- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

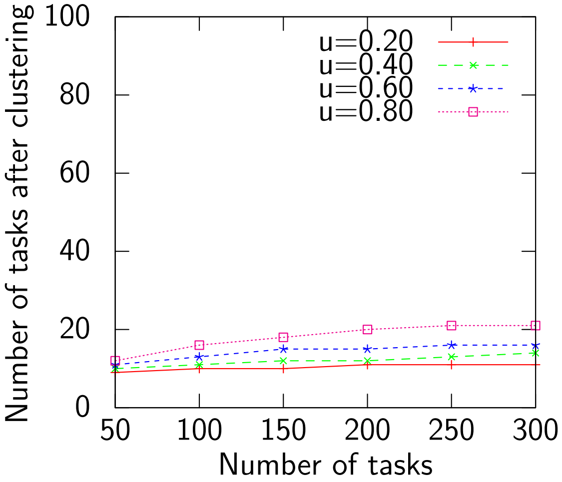
A greedy heuristic

- **Idea**: Successive clusterings from an initial task set
- Heuristic cost function based on response time R computation: $h(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \frac{R_n}{D_n}$ ($\frac{R_n}{D_n}$ closer to 1 means less margin for the scheduler).

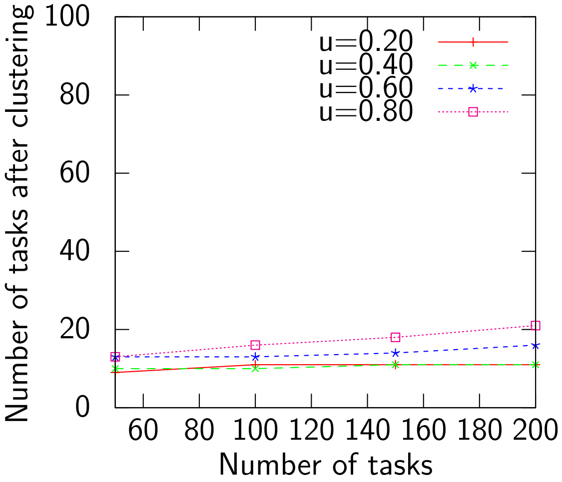


- Sustainable unschedulability: a task set deemed unschedulable remains so after clustering $\Rightarrow$ avoid useless search

Results under DM



Results under EDF



Outline

Conclusion

We presented in this talk:

- the task clustering problem,
- its complexity,
- some heuristic principles,
- and a first heuristic.

Current and future work:

- adding precedences between tasks
- then applying task clustering to multi-processor systems