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

Introduction

Definition

Complexity

Solution

Conclusion
Outline

Introduction

Definition

Complexity

Solution

Conclusion
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
    ⇒ increase the risk of cache misses (larger WCET)

- **Memory**
  - Task level: one stack by task
  - Scheduler level: increase in the number of priorities required
    ⇒ number of preemption increases
    ⇒ execution stack grows
Task Clustering

RTOS limitations

Having numerous threads:

- **Scheduling overhead**
  - Scheduler level: handle large queues
  - Increase of context switches
    ⇒ increase the risk of cache misses (larger WCET)

- **Memory**
  - Task level: one stack by task
  - Scheduler level: increase in the number of priorities required
    ⇒ number of preemption increases
    ⇒ execution stack grows

→ Several functionalities grouped together in a thread
Manually made in industry (error prone, tedious)
Outline

Introduction

Definition

Complexity

Solution

Conclusion
System model used

Program = a set of tasks $\tau_i$:

- $T_i$: 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** \( \tau_i \) and \( \tau_j \) into \( \tau_{ij} \) with \( D_i \leq D_j \)

\[
\begin{align*}
C_{ij} &= C_i + C_j \\
T_{ij} &= T_i = T_j \quad \text{(by restriction we only regroup tasks with equal periods)}
\end{align*}
\]

- 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 \( \tau_{i,j} \) in that order

(a) Initial system with tasks \( \tau_i, \tau_x \) et \( \tau_j \)

(b) System after clustering \( \tau_i \) with \( \tau_j \)
Cluster model: Deadline choice for the cluster

- **Case 1:** Maximum deadline $D_j$
  - if $(D_j - C_j \leq D_i)$
  - and $\tau_{i,j}$ in that order

![Diagram](attachment:cluster_model_diagram.png)

(a) Initial system with tasks $\tau_i, \tau_x$ et $\tau_j$

(b) System after clustering $\tau_i$ with $\tau_j$

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 \( \tau_{i,j} \) \( \tau_i \| \tau_j \) in that order

(a) Initial system with tasks \( \tau_i, \tau_x \) et \( \tau_j \)

(b) System after clustering \( \tau_i \) with \( \tau_j \)

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
  - $\tau_{ij} \tau_i \tau_j$ or $\tau_{ij} \tau_j \tau_i$ (order does not matter)

(a) Initial system with tasks $\tau_i, \tau_x$ et $\tau_j$

(b) System after clustering $\tau_i$ with $\tau_j$

- 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
  - $\tau_{i,j}$ $\tau_i \tau_j$ or $\tau_{i,j}$ $\tau_j \tau_i$ (order does not matter)

- System may become unschedulable after clustering
  - Schedulability must be checked after each non zero-cost clustering!
Cluster model
Valid cluster

**Theorem**

Let $S$ be a task set and $\Phi$ be a priority assignment. Let $S'$ the task set $S$ after clustering of two tasks $\tau_i$ and $\tau_j$. $S'$ is schedulable under $\Phi \Rightarrow S$ is schedulable under $\Phi$.

However, the converse is not always true.

**Definition**

The clustering is valid iff schedulability is preserved after clustering
Outline

Introduction

Definition

Complexity

Solution

Conclusion
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 $\tau_a, \tau_b, \tau_c$ et $\tau_d$:

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

→ Motivates to work towards a **heuristic**
Outline

Introduction

Definition

Complexity

Solution

Conclusion
Heuristic principles

• Classical optimization techniques are based on cost functions to choose “promising candidates”
  ⇒ 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: $h(S) = \sum_{n=1}^{\mid S \mid} \frac{R_n}{D_n}$

($\frac{R_n}{D_n}$ 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: 
$$h(S) = \sum_{n=1}^{\mid S \mid} \frac{R_n}{D_n}$$

($\frac{R_n}{D_n}$ 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: $h(S) = \sum_{n=1}^{\left|S\right|} \frac{R_n}{D_n}$

  ($\frac{R_n}{D_n}$ 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:
\[
h(S) = \sum_{n=1}^{\mid S \mid} \frac{R_n}{D_n}
\]

\( \frac{R_n}{D_n} \) 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 $R$

\[
h(S) = \sum_{n=1}^{\lvert S \rvert} \frac{R_n}{D_n}
\]

($\frac{R_n}{D_n}$ 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 $R$

Computational cost: $h(S) = \sum_{n=1}^{\mid S \mid} \frac{R_n}{D_n}$

($\frac{R_n}{D_n}$ 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 $R$

Cost computation:

$$h(S) = \sum_{n=1}^{\left|S\right|} \frac{R_n}{D_n}$$

($\frac{R_n}{D_n}$ 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: $h(S) = \sum_{n=1}^{\vert S \vert} \frac{R_n}{D_n}$

  ($\frac{R_n}{D_n}$ 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 $R$

$$h(S) = \sum_{n=1}^{\vert S \vert} \frac{R_n}{D_n}$$

($\frac{R_n}{D_n}$ 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:** $h(S) = \sum_{n=1}^{\vert S \vert} \frac{R_n}{D_n}$

($\frac{R_n}{D_n}$ 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 $R$

Heuristic cost function computation:

$$h(S) = \sum_{n=1}^{\left|S\right|} \frac{R_n}{D_n}$$

($\frac{R_n}{D_n}$ 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 $R$

Heuristic cost function computation: 

$$h(S) = \sum_{n=1}^{\left|S\right|} \frac{R_n}{D_n}$$

($\frac{R_n}{D_n}$ 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 $R$

Heuristic cost function computation: \( h(S) = \sum_{n=1}^{\mid S \mid} \frac{R_n}{D_n} \)

(\( \frac{R_n}{D_n} \) 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 $R$

Heuristic cost function computation:

$$h(S) = \sum_{n=1}^{\left| S \right|} \frac{R_n}{D_n}$$

($\frac{R_n}{D_n}$ 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 $R$

    \[
    h(S) = \sum_{n=1}^{\lvert S \rvert} \frac{R_n}{D_n}
    \]

    ($\frac{R_n}{D_n}$ 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 $R$

Cost computation: $h(S) = \sum_{n=1}^{\lfloor S \rfloor} \frac{R_n}{D_n}$

($\frac{R_n}{D_n}$ 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 $R$

Heuristic cost function computation: $h(S) = \sum_{n=1}^{S} \frac{R_n}{D_n}$

($\frac{R_n}{D_n}$ 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 $R$

$$h(S) = \sum_{n=1}^{\left|S\right|} \frac{R_n}{D_n}$$

($\frac{R_n}{D_n}$ 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: $h(S) = \sum_{n=1}^{\mid S \mid} \frac{R_n}{D_n}$

  ($\frac{R_n}{D_n}$ 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 $R$

$$h(S) = \sum_{n=1}^{\mid S\mid} \frac{R_n}{D_n}$$

($\frac{R_n}{D_n}$ 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 \( R \)

\[
h(S) = \sum_{n=1}^{\left|S\right|} \frac{R_n}{D_n}
\]

\( \frac{R_n}{D_n} \) 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 $R$

Cost function computation:

$$ h(S) = \sum_{n=1}^{\vert S \vert} \frac{R_n}{D_n} $$

($\frac{R_n}{D_n}$ 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

Figure: Task clustering under Deadline Monotonic scheduling algorithm (u = processor utilization factor)
Results under EDF

Figure: Task clustering under EDF scheduling algorithm (u = processor utilization factor)
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