Minimizing the cardinality of a real-time task set by automated task clustering

Antoine Bertout, Julien Forget and Richard Olejnik

Laboratoire d'Informatique Fondamentale de Lille (LIFL)
Université Lille1, France
firstname.lastname@lifl.fr

October 16, 2013

Context

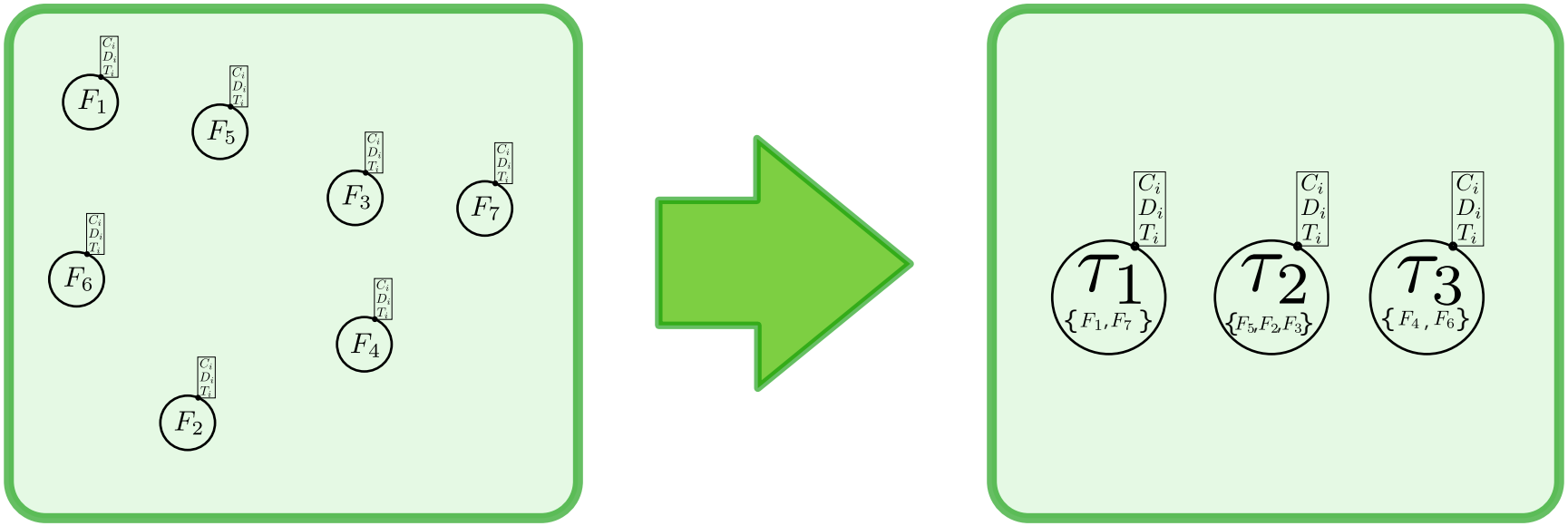
- Real-time systems with uniprocessor
- Task model (Liu et Layland)
  - C_i: worst case execution time of τ_i
  - T_i: activation period of τ_i
  - D_i: deadline of τ_i
  - constraint deadlines: $D_i \leq T_i$
  - independent and synchronous tasks

Problem

- Up to ≈ 1000 **high level functionalities** in RT system software (e.g. aileron command, read pressure sensor, etc.)
- 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)
  $\rightarrow$ Several functionalities grouped together in a thread
- Usually hand made in industry (error prone, tedious)
- **Our solution**: automated task clustering

Objective

- Automatically grouping functionalities to tasks to minimize their number
- while keeping the system **functionally equivalent**
- while **preserving schedulability**

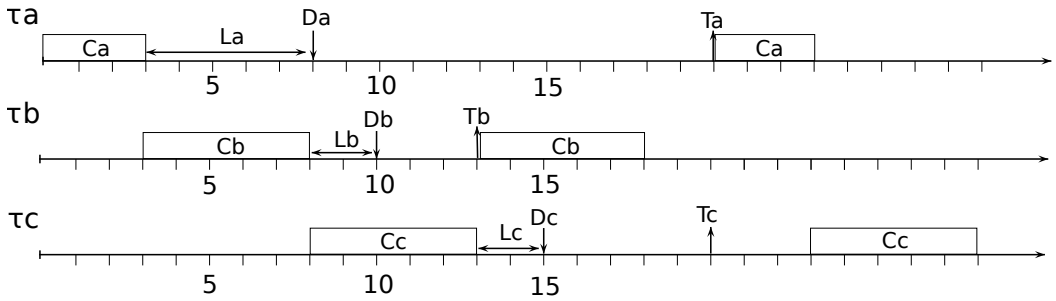


Task clustering

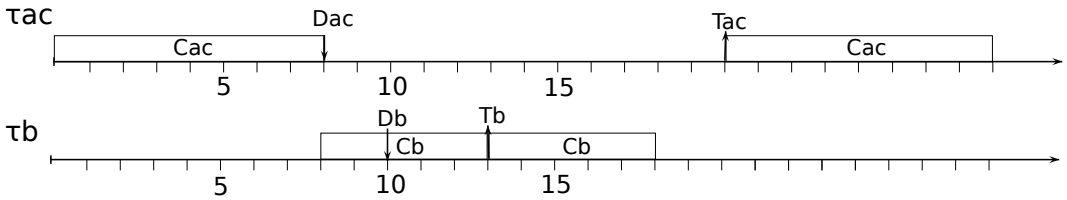
- **Cluster** τ_i and τ_j into τ_{ij}
  - $C_{ij} = C_i + C_j$
  - $T_{ij} = T_i = T_j$
  - $D_{ij} = min(D_i, D_j)$ (taking shortest deadline ensures respect of initial constraints)
- What is a **valid cluster**?
  1. $T_i = T_j$
  2. Laxity $L_i = (D_i - C_i) \geq C_y$ (if $D_i \geq D_j$)
  3. Whole system is still schedulable

Schedulability problem

- System may not be still schedulable after clustering



(a) Initial system with tasks τ_a,τ_b et τ_c



(b) System after task clustering τ_a et τ_c

- In the second diagram, τ_b **misses its first deadline** after clustering of τ_a and τ_c

$\rightarrow$ Schedulability after each clustering must be checked!

Task clustering complexity

- **Combinatorial explosion**: number of possible clusterings in the Bell number range (e.g.,$\mathbf{B_{500} = 10^{844}}$)
- *Exact schedulability tests* have often pseudo-polynomial complexity
  Sufficient tests a linear complexity
- Exhaustive search untractable even using linear sufficient tests (no response after several days of computation for 20 tasks from first experiments)

$\rightarrow$ We need a **heuristic** to tackle this task clustering

Heuristic Approach

- I would be happy to explain to you how we use *schedulability test* as *heuristic cost function* in the front of my **poster**!
- Thanks for your attention!