Partition Scheduling in Distributed Integrated Modular Avionics

João Miguel Fonseca Gonçalves

joaomfgoncalves@tecnico.ulisboa.pt

Instituto Superior Técnico

November 29, 2019

Outline

1. Motivation
   - Avionic Architectures
   - Partition Schedules
   - Objectives
2. Partition Scheduling Model
   - Variables
   - Distribution Constraints
   - Communication Constraints
   - Timing Constraints
   - Optimization
3. Methodology
   - Mixed Integer Linear Programming
   - Heuristic optimization
   - Global Optimization
4. Results
   - Algorithmic Performance
   - Scheduling Tool
5. Conclusions

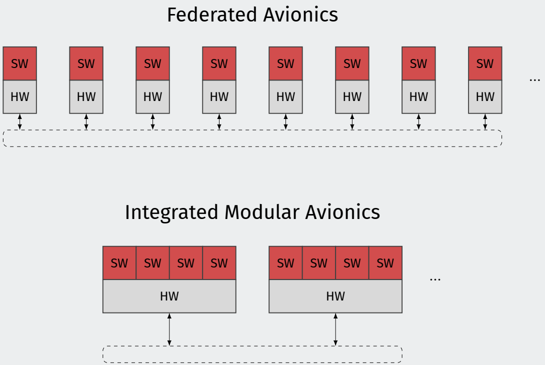
Avionic Architectures

Federated Avionics

- Dedicated hardware for each application

IMA

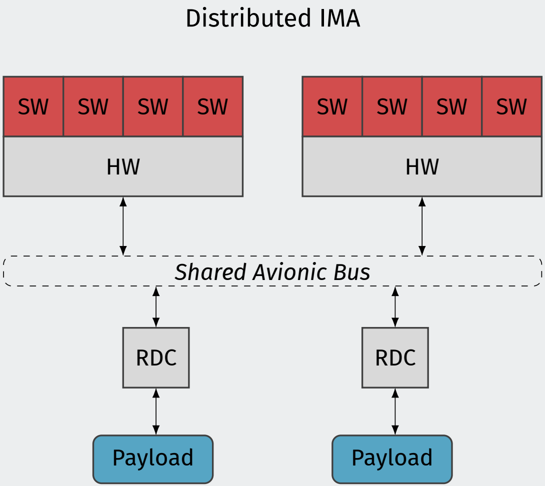
- Resource sharing between unrelated applications
- Active research area



Distributed Integrated Modular Avionics

Benefits:

- Maintains safety
- Efficient usage of resources
- Reduced space, weight and power
- Uses open standards
- Enables the involvement of smaller players



Strong Partitioning

How is this achievable?

Space Partitioning

- Statically isolated memory for each application

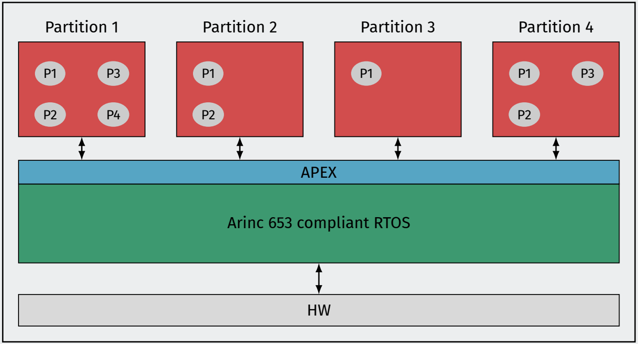
Time Partitioning

- Guaranteed processor time for each application
- Deterministic scheduling

Arinc 653

- Avionics Application Software Standard Interface

Core Processing Module



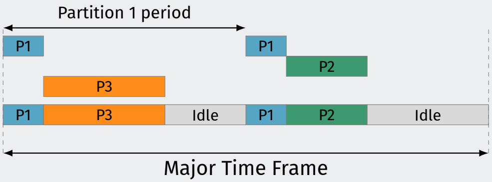
Partition Schedules

Characteristics

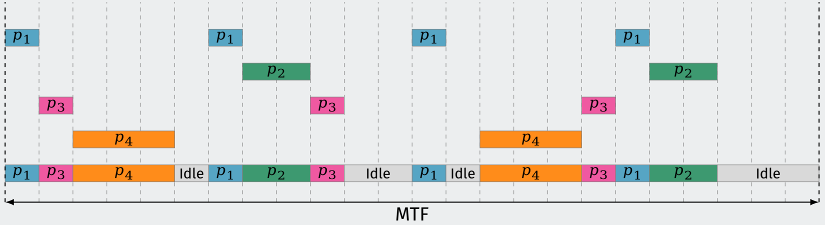
- Static
- Strictly periodic
- Functional requirements:
  - Inter-partition communication
  - Access to resources
  - Redundancy configuration

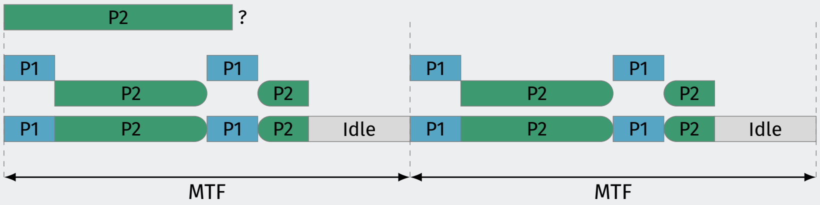
Status

- Manual, iterative process
- *In-house* solutions



Partition Schedules – Examples





Objectives

Challenges:

- NP-completeness
- Defining the system requirements
- Reconfiguration is lengthy and expensive

Objectives:

- Comprehensive mathematical description
- Automate schedule generation and validation
- Find flexible solutions

Partition Scheduling Model

1 Motivation
2 Partition Scheduling Model
3 Methodology
4 Results
5 Conclusions

The Partition Scheduling Problem

Schedule N_p partitions:

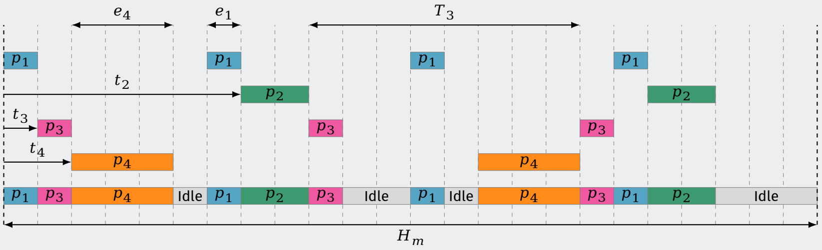
- T_i – partition i period
- e_i – partition i execution time (WCET)
- s_i – partition i memory requirement

in N_c modules:

- S_m – module m memory capacity

Find, for each partition i:

- f_i – assigned module
- t_i – starting offset

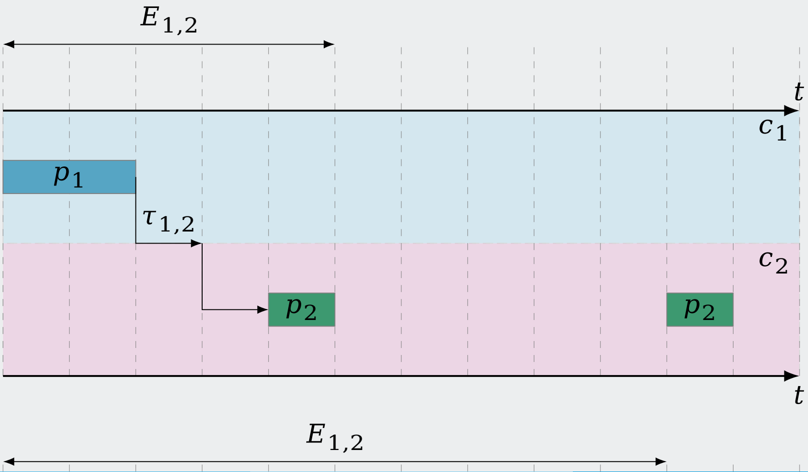


Distribution Constraints

- **Domain** – partitions can be scheduled in only some modules
- **Exclusion** – some partitions must be scheduled in distinct modules
- **Inclusion** – some partitions must be scheduled in the same module
- **Memory** – a module's memory must not be exceeded
- **Uniqueness** – a partition can only be scheduled in one module

Inter-partition Communication

- $E_{i,j}$ – communication 'chains'
- $\tau_{m,n}$ – inter-module communication delay
- Synchronous communication



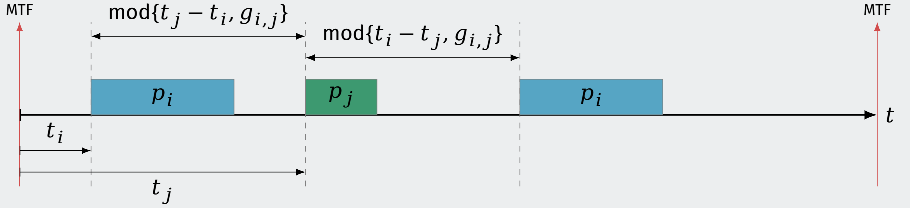
Timing Constraints

- No temporal overlap between two partitions in the same module

Theorem [1]:

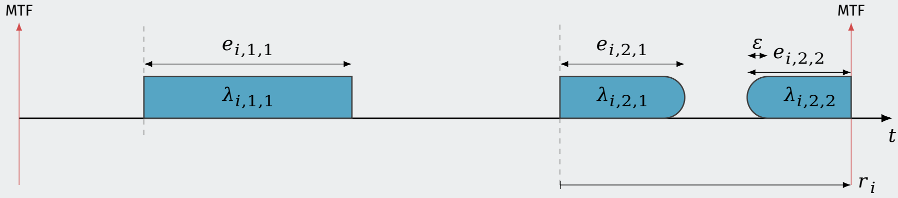
$$g_{i,j} = gcd\left\{T_i, T_j\right\}$$

$$e_i \leq mod\left\{t_j - t_i, g_{i,j}\right\} \leq T_j - e_j$$



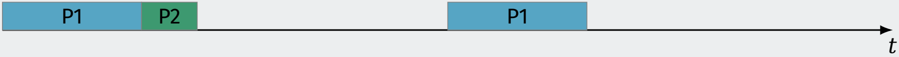
Timing Constraints

- Multiple windows of execution require an execution penalty, ε
- Response time, r_i, is bounded

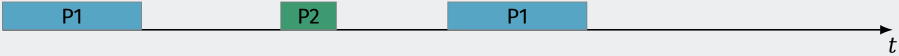


Optimizing Potential Execution

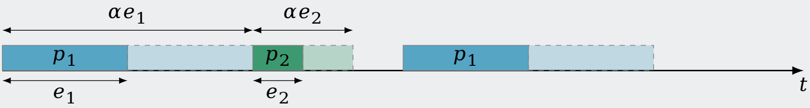
- Partition 1 cannot increase its execution



- It is possible to increase execution times without rescheduling



Optimizing Potential Execution



Objective: maximize α

- α – ‘Minimum factor that scales all execution windows without overlaps’

Partition Scheduling Model

1 Motivation

2 Partition Scheduling Model

3 Methodology

4 Results

5 Conclusions

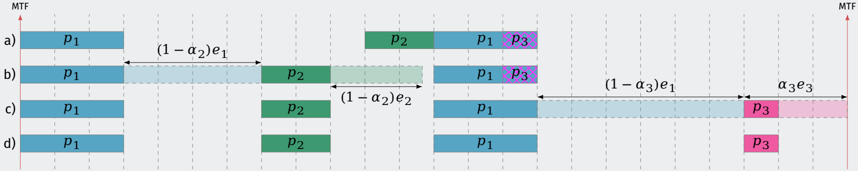
Mixed Integer Linear Program

- Optimization problem with both integer and real-valued variables
- Branch and bound / branch and cut algorithms
- Yields an optimal solution

$$
\begin{aligned}
\max \quad & \alpha \\
\text{s.t.} \quad & 0 \le \alpha \le \min_{p_i \in P} \left\{ \frac{T_i}{e_i} \right\} \\
& \forall p_i \in P: \quad \sum_{c_m \in C} a_{i,m} = 1 \\
& \qquad 0 \le t_i \le T_i - e_i \\
& \qquad \forall c_m \in \{C \setminus D_i\}: \ a_{i,m} = 0 \\
& \forall c_m \in C: \quad \sum_{p_i \in P} a_{i,m} s_i \le S_m \\
& \forall p_i, p_j \in P^2: \quad f_i \neq f_j, \ \forall c_m \in C: \ a_{i,m} \le a_{j,m} \\
& \qquad f_i = f_j, \ \forall c_m \in C: \ a_{i,m} = a_{j,m} \\
& \forall p_i, p_j \in P, j > i: \forall c_m \in C: \ t_j - t_i - q_{i,j} g_{i,j} \ge \alpha e_i - \\
& \qquad - Z \left(2 - a_{i,m} - a_{j,m}\right) \\
& \qquad \forall c_m \in C: \ t_j - t_i - q_{i,j} g_{i,j} \le -g_{i,j} \alpha e_j - \\
& \qquad - Z \left(2 - a_{i,m} - a_{j,m}\right) \\
& \qquad \frac{e_i - T_i}{g_{i,j}} \le q_{i,j} \le \frac{T_j - e_i}{g_{i,j}} \\
& \qquad 0 \le t_j - t_i - q_{i,j} g_{i,j} \le g_{i,j} \\
& \forall p_i, p_j \in P, E_{ij}^{max} < \infty: \quad t_j - t_i - q_{i,j} g_{i,j} + e_j + x_{i,j} T_j \le E_{ij}^{max} \\
& \qquad t_j - t_i - q_{i,j} g_{i,j} + e_i - \sum_{c_m, c_n \in C} \left(y_{i,j,m,n} \tau_{m,n} \right) + \\
& \qquad + x_{i,j} Z \ge 0 \\
& \qquad \forall c_m, c_n \in C: \ y_{i,j,m,n} \ge a_{i,m} + a_{j,n} - 1 \\
& \qquad \forall c_m, c_n \in C: \ y_{i,j,m,n} \le a_{i,m} \\
& \qquad \forall c_m, c_n \in C: \ y_{i,j,m,n} \le a_{j,n} \\
& a_{i,m} \in \{0, 1\}, \ t_i \in \mathbb{Z}, \ q_{i,j} \in \mathbb{Z} \\
& x_{i,j} \in \{0, 1\}, \ y_{i,j,m,n} \in \{0, 1\}
\end{aligned}
$$

Best Response Algorithm [3]

- Based on the Game Theory algorithm of the same name
- Partitions sequentially move to the best available offset
- When no partition can improve its α value, equilibrium is reached
- Optimality dependant on the starting point
- Time complexity: $\mathcal{O}(N \cdot T^{N-1})$



Global Optimization

Explore different distributions of partitions among modules

- Optimize each module's schedule using the previous algorithms
- Change the distribution and repeat

Stochastic Optimization Algorithms

Operators:

- Move partitions between two modules
- Swap partitions between two modules
- Shuffle partition offsets
- Add/remove execution windows

Meta-heuristic algorithms:

- Simulated Annealing
- Tabu Search
- Genetic Algorithm

Partition Scheduling Model

1 Motivation

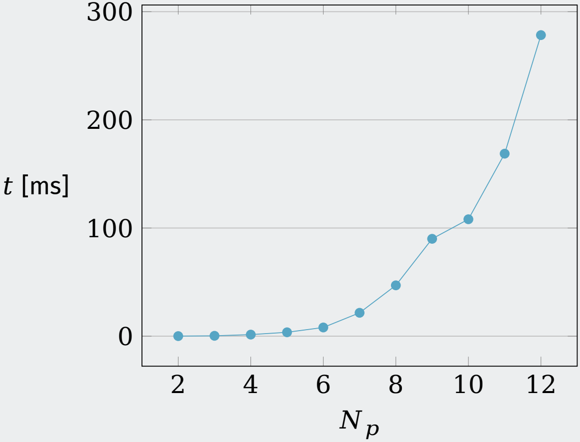
2 Partition Scheduling Model

3 Methodology

4 **Results**

5 Conclusions

Best Response Algorithm



Test Cases

Designation	Modules	Partitions	Chains	α_{best}
$2M6P$	2	6	0	5.5*
$4M10P$	4	10	3	6.403
$4M20P$	4	20	8	2.875
$8M40P$	8	40	15	2.984
$20M100P$	20	100	40	2.325
$3M15P$-S	3	15	3	1.26

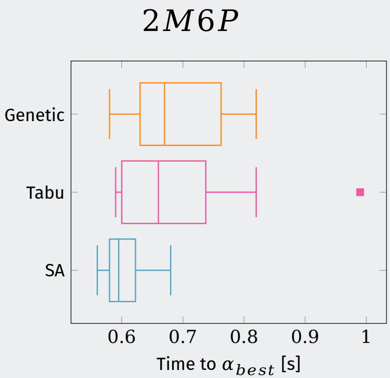
$$T \in \{100, 200, 500, 1000\}$$

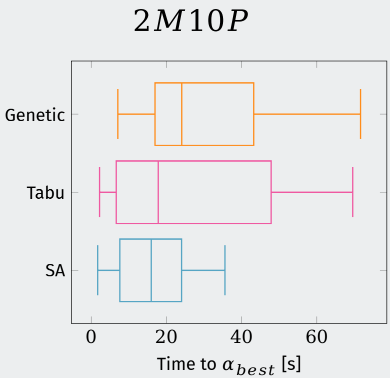
Feasible Solution

Instance	t_{MILP}	$t_{heuristic}$ (median)
$2M6P$	1.00 s	0.593 s
$4M10P$	1.34 s	0.595 s
$4M20P$	6.29 s	0.830 s
$8M40P$	310.7 s	1.155 s
$20M100P$	$>$ 24 h	23.32 s
$3M15P$-S	NA	74.53 s

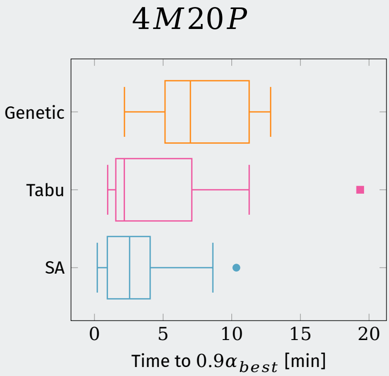
Improved solution

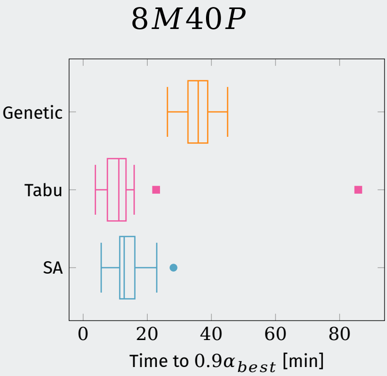
MILP solver does not converge in under 24 h, except for $2M6P$.



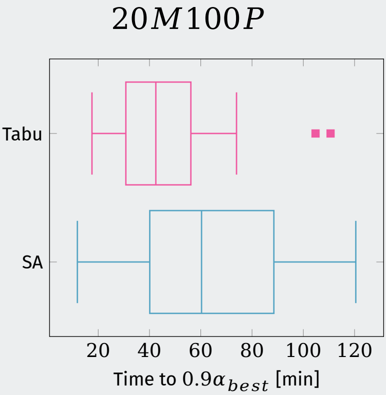


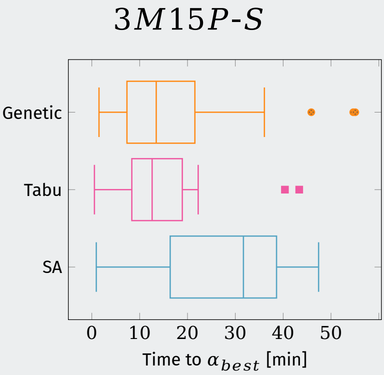
Improved solution





Improved solution





Achievements

- A scheduling tool and framework that supports a variety of constraints
- Capable of producing flexible solutions in moderate amounts of time
- A mathematical model that yields optimal solutions when time is not an issue

Recommendations for Future Work

- Integration with other (D)IMA configuration or V&V frameworks
- Partition scheduling in multicore (D)IMA systems

Selected Bibliography

[1] J.H.M. Korst. ‘Periodic multiprocessor scheduling’. English. PhD thesis. Technische Universiteit Eindhoven, Department of Mathematics and Computer Science, 1992. DOI: 10.6100/IR388787.

[2] Slawomir Samolej. ‘ARINC Specification 653 Based Real-Time Software Engineering.’. In: *e-Informatica* 5.1 (2011), pp. 39–49.

[3] Ahmad Al Sheikh, Olivier Brun, Pierre-Emmanuel Hladik and Balakrishna J Prabhu. ‘Strictly periodic scheduling in IMA-based architectures’. In: *Real-Time Systems* 48.4 (2012), pp. 359–386. DOI: 10.1007/s11241-012-9148-y.

[4] Clément Pira and Christian Artigues. ‘Line search method for solving a non-preemptive strictly periodic scheduling problem’. In: *Journal of Scheduling* 19.3 (2016), pp. 227–243. DOI: 10.1007/s10951-014-0389-6.