50 YEARS OF COMPUTER SCIENCE AT YORK

UNIVERSITY of York

www.york.ac.uk/computerscience

HISE Group

RTS York

Ada europe

Resilience-aware Mixed-criticality DAG Scheduling on Multi-cores for Autonomous Systems

Authors: Jie Zou, Xiaotian Dai, John A. McDermid

Department of Computer Science, University of York

Email: jie.zou@York.ac.uk, xiaotian.dai@York.ac.uk, john.mcdermid@york.ac.uk

Abstract: Advanced driver-assistance and semi-autonomous systems represent the next major computing demand for road vehicles, which are complex and safety-critical with strict real-time and resource constraints and have a deep processing pipeline with strong dependencies between different functions. Further, tasks with different criticalities share the same hardware. This work proposed a novel mixed-criticality DAG-based multi-core static scheduling method considering low critical tasks' survivability and precedence constraints between tasks with different criticalities. This produces a ***consistent schedule for different system modes enabling task-level mode change and improving the resilience of the system***.

1. Problems of Existing Static Scheduling Methods

- Most static scheduling work considering task dependencies does not consider the survivability of low criticality tasks.
- The schedules for high and low modes are different. More efforts are needed to check the safety of schedules during mode change.

2. Contribution of This Work

- Generates one consistent schedule considering the survivability of LO tasks to realise task-level mode change and significantly improves system resilience.
- Reduces the complexity of task-level mode change, which can accelerate the recovery of specific impacted LO task.

3. Mixed-Criticality Task Model

We adopt dual-criticality system considering criticality-dependent WCET estimation (i.e., for HI critical tasks C(LO) < C(HI)). A task τ_i can be defined by the tuples (T_i, D_i, C_i(HI), C_i(LO), L_i).

4. Consistent Mixed-Criticality DAGs Scheduling

Start from the last time point of the hyperperiod and the last layer of DAGs in the system. All HI critical tasks can be executed as late as possible and scheduled start time will be kept the same in both system modes.

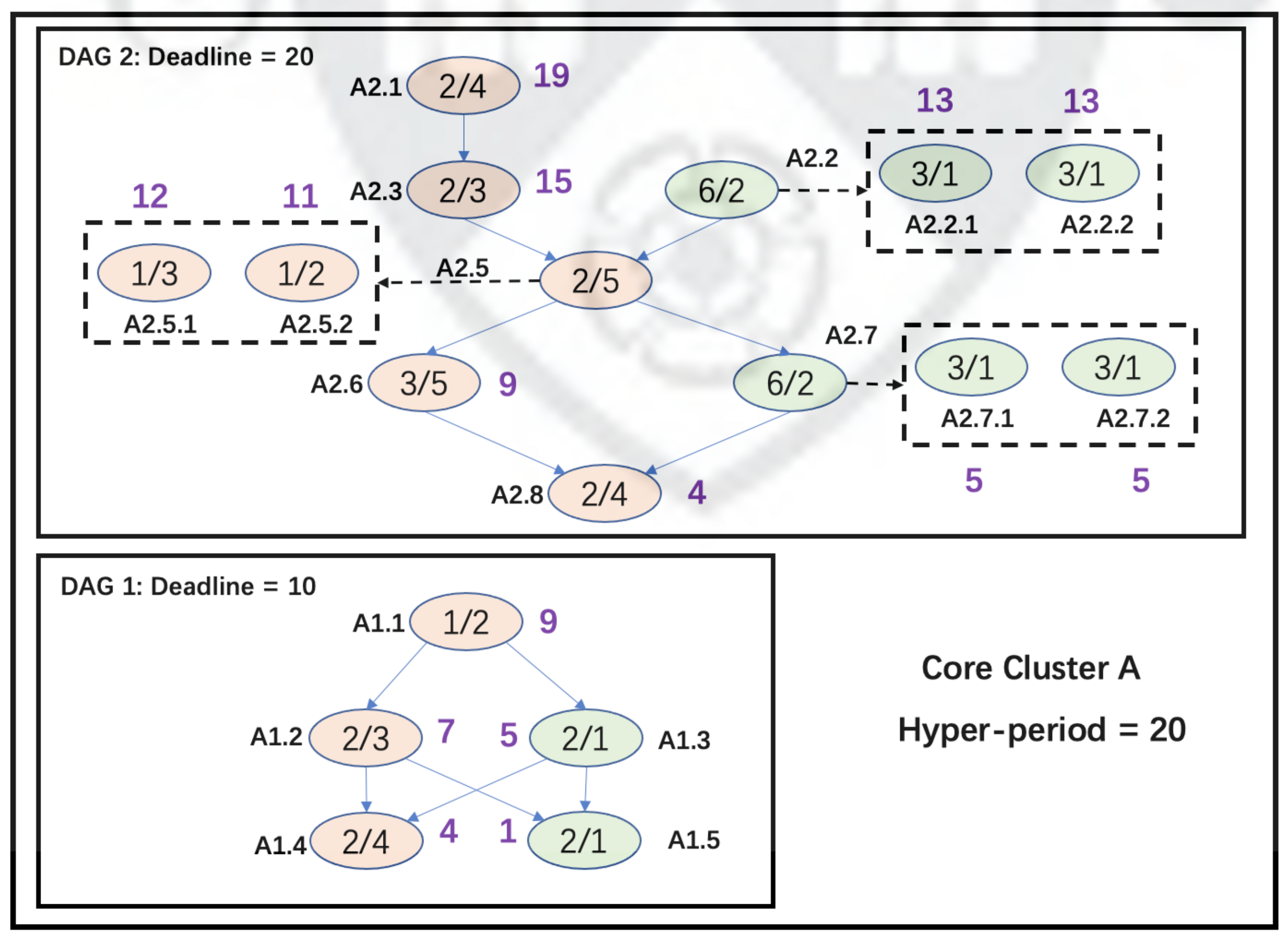

Fig. 1: Mixed-criticality DAGs example (Green nodes: LO-tasks; Red nodes: HI-tasks)

4.1 Schedule Calculation

Step 1. Schedule calculation in HI mode:

The schedule should be generated based on the HI mode task behaviour to guarantee the execution of HI tasks. The existence of LO criticality tasks can assist in anchoring the target schedule region of LO tasks in LO mode.

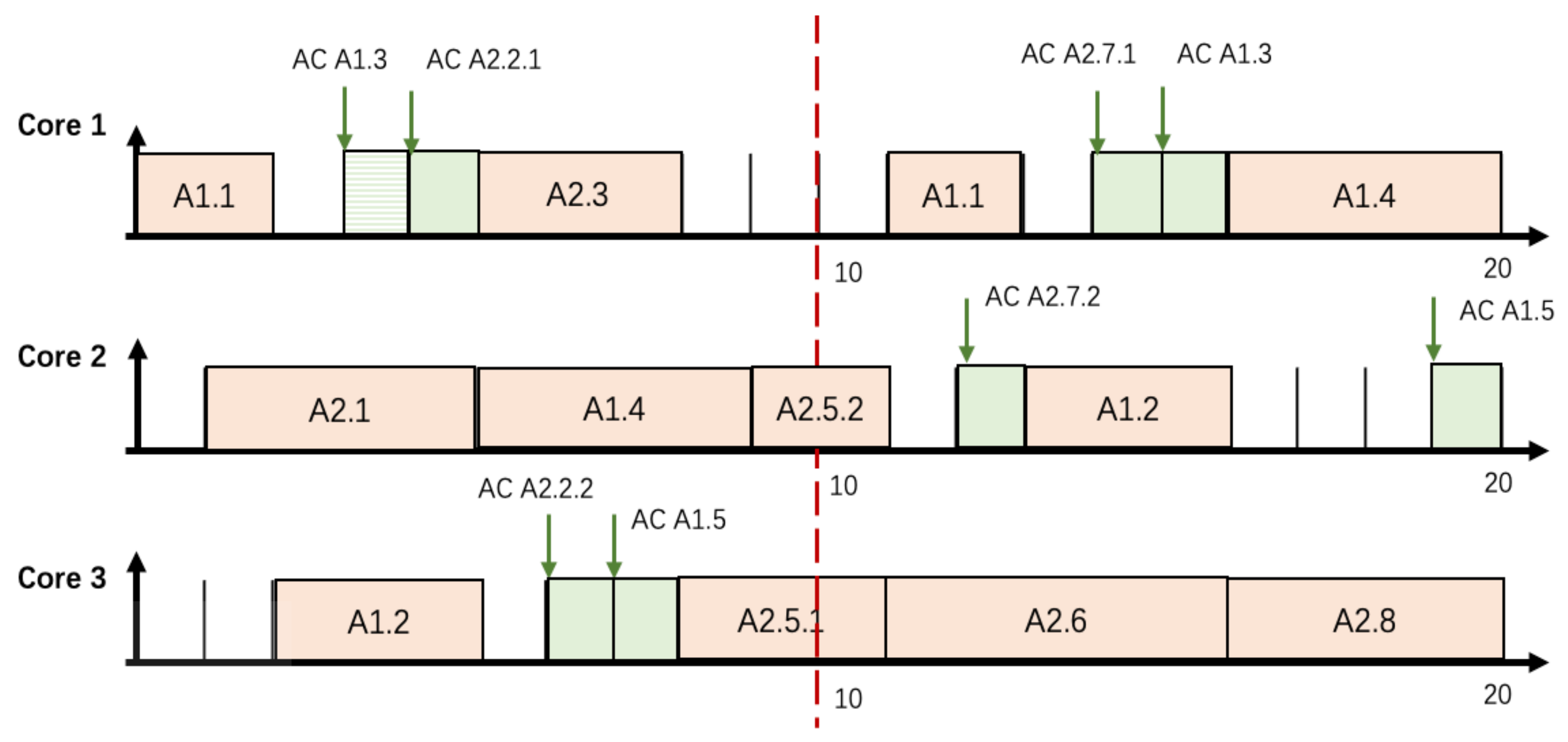

Fig. 2: The schedule generated by step 1

Step 2. Consistent schedule generation:

Based on the schedule calculated in step 1, all tasks are performed with LO mode behaviour in this step. The execution time of HI tasks is shortened, and more time is freed to schedule LO tasks.

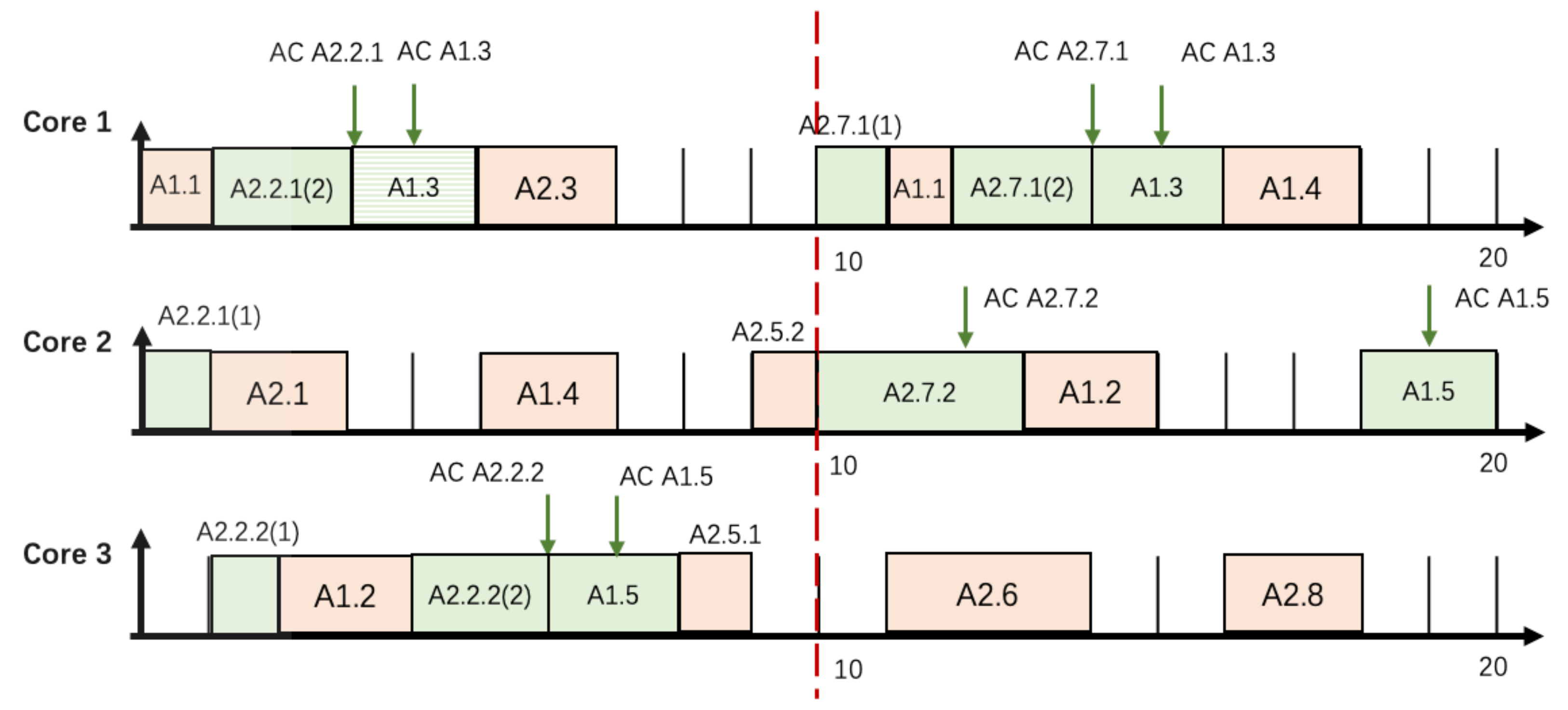

Fig. 3: The consistent schedule from step 2

5. Future Work Plan

The benefits brought by our method will be further verified based on large scale simulation. Furthermore, it will be applied to more realistic examples – as we are doing with a mobile delivery robot.

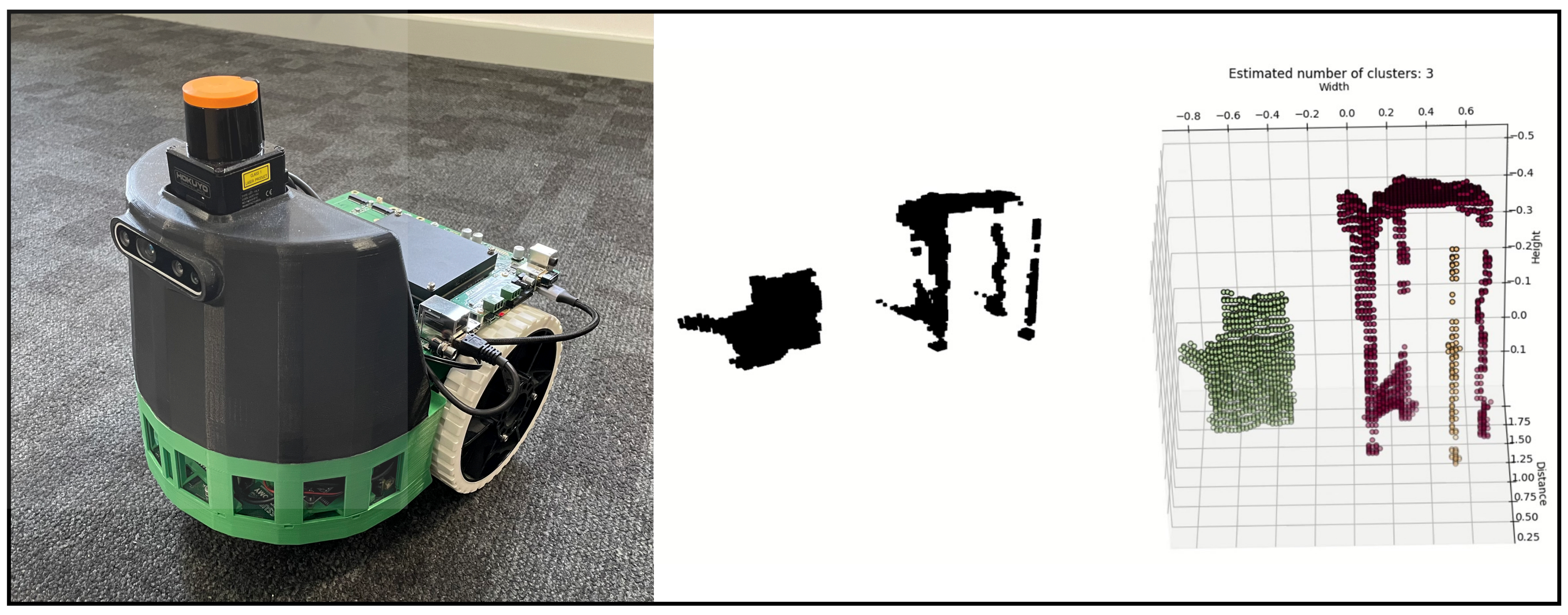

Fig. 4: ISA Robot from AAIP[1] Project

[1] **Assuring Autonomy International Programme (AAIP)** https://www.york.ac.uk/assuring-autonomy/about/