Max
Planck
Institute
for
Software Systems

Offline Equivalence:

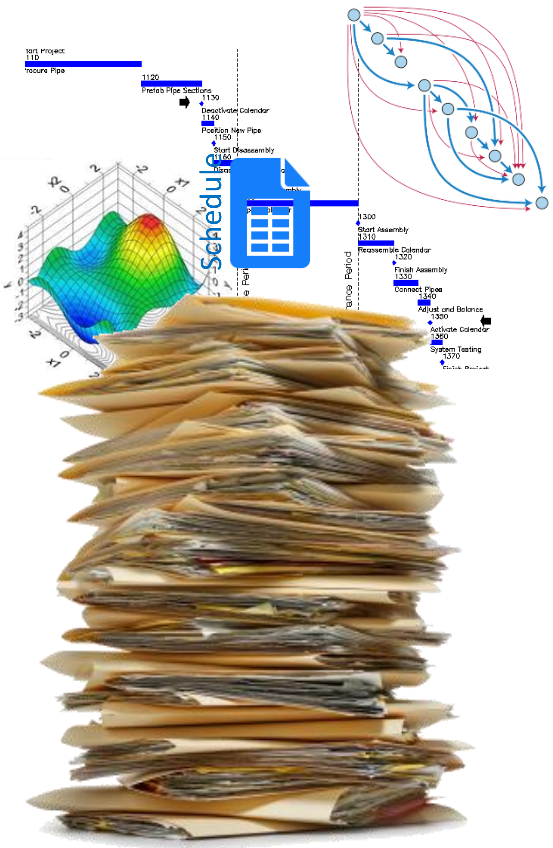
A Non-Preemptive Scheduling Technique for Resource-Constrained Embedded Real-Time Systems

Mitra Nasri* Björn B. Brandenburg

MPI-SWS, Kaiserslautern, Germany

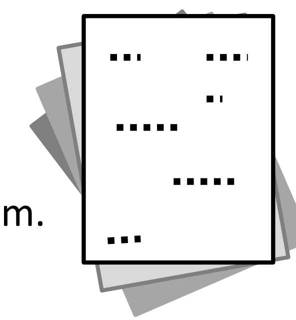
RTAS, April 2017

What do you do if
you have a nice scheduling table
that doesn't fit into memory?



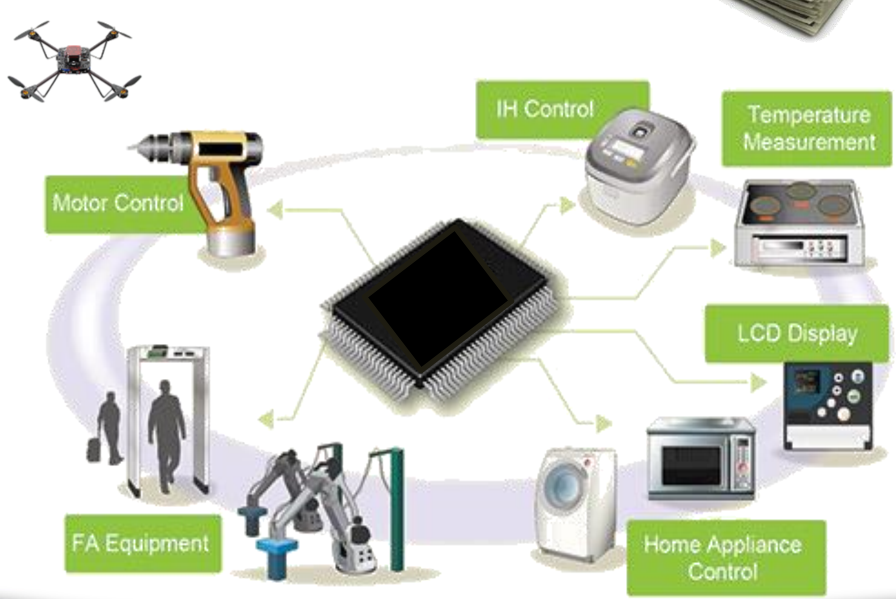
Offline Equivalence

allows you to store only a little "crucial" information
to **rebuild** your table at **runtime**
with the help of an efficient **online scheduling** algorithm.



Motivation

- Many embedded systems (still) have limited processing power and memory
- Usually no operating system
- Naturally **non-preemptive**



Arm Cortex MCU family

STM32 32-bit ARM Cortex MCUs | < | STM32F2 Series | STM32F3 Series | STM32F4 Series | STM32F7 Series | S

Total Parts: (752) for STM32 32-bit ARM Cortex MCUs | Matching Parts : (90)

Part Number	Package	Core	Operating Frequency (MHz) (Processor speed)	FLASH Size (kB) (Prog)	Internal RAM Size (kB)	I/Os (High Current)
STM32L011G4	UFQFPN 28 4x4 x0.55	ARM Cortex-M...	32	16	2	24
STM32L011K4	LQFP 32 7x7x1.4, ...	ARM Cortex-M...	32	16	2	28
STM32L021D4	TSSOP 14	ARM Cortex-M...	32	16	2	11
STM32L021F4	UFQFPN 20 3x3 x0.6	ARM Cortex-M...	32	16	2	16
STM32L021G4	UFQFPN 28 4x4 x0.55	ARM Cortex-M...	32	16	2	24
STM32L021K4	LQFP 32 7x7x1.4	ARM Cortex-M...	32	16	2	28
STM32L031F4	TSSOP 20	ARM Cortex-M...	32	16	8	15
STM32L071C8	LQFP 48 7x7x1.4	ARM Cortex-M...	32	64	20	37
STM32L071RZ	LQFP 64 10x10 x1.4, ...	ARM Cortex-M...	32	192	20	51
STM32L071VB	LQFP 100 14x14x1.4	ARM Cortex-M...	32	128	20	84

Existing Approaches

Table-driven scheduling or cyclic executive

Online scheduling

- Low runtime overhead
- High schedulability ratio
- Flexible: allows adding constraints during construction

Table must be stored in memory

Offline Equivalence

- ~~No need to store a table~~
- Stores less information

Less flexibility to add complex constraints

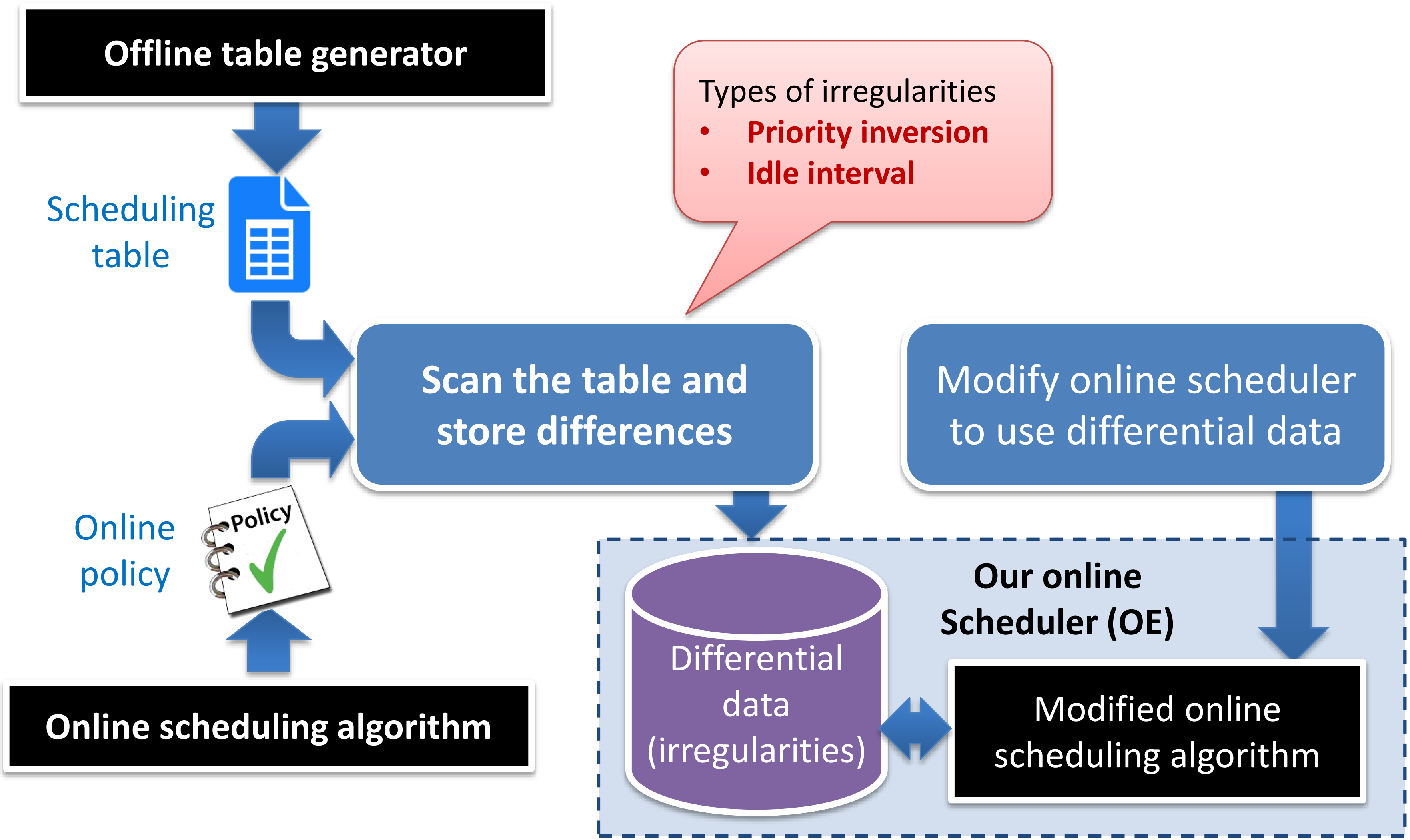
work-conserving
(fixed-priority, EDF, etc.)

Non-work-conserving
(Precautious-RM, CW-EDF, etc.)

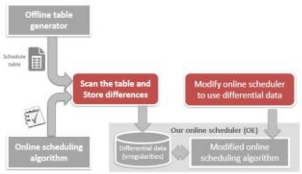
A power train ECU [Anssi13]:
- 6 periodic tasks with release offset
- Periods {1, 5, 10, 10, 40, 100}
- 500 jobs in a hyperperiod
- Offline table is at least 2 KiB

An automotive benchmark from Bosch [Kramer15]:
- Periods are {1, 2, 5, 10, 20, 50, 100, 200, 1000}
- 1886 jobs in a hyperperiod
- Adding a functionality with 30 frames per second leads to 63,238 jobs in a hyperperiod

This Paper: Offline Equivalence



Contributions

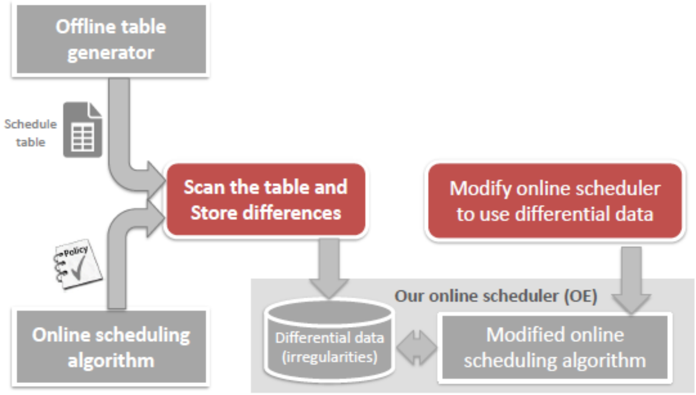


- **Offline equivalence technique**

- **An efficient offline table generation algorithm**
  (for a non-preemptive set of jobs)

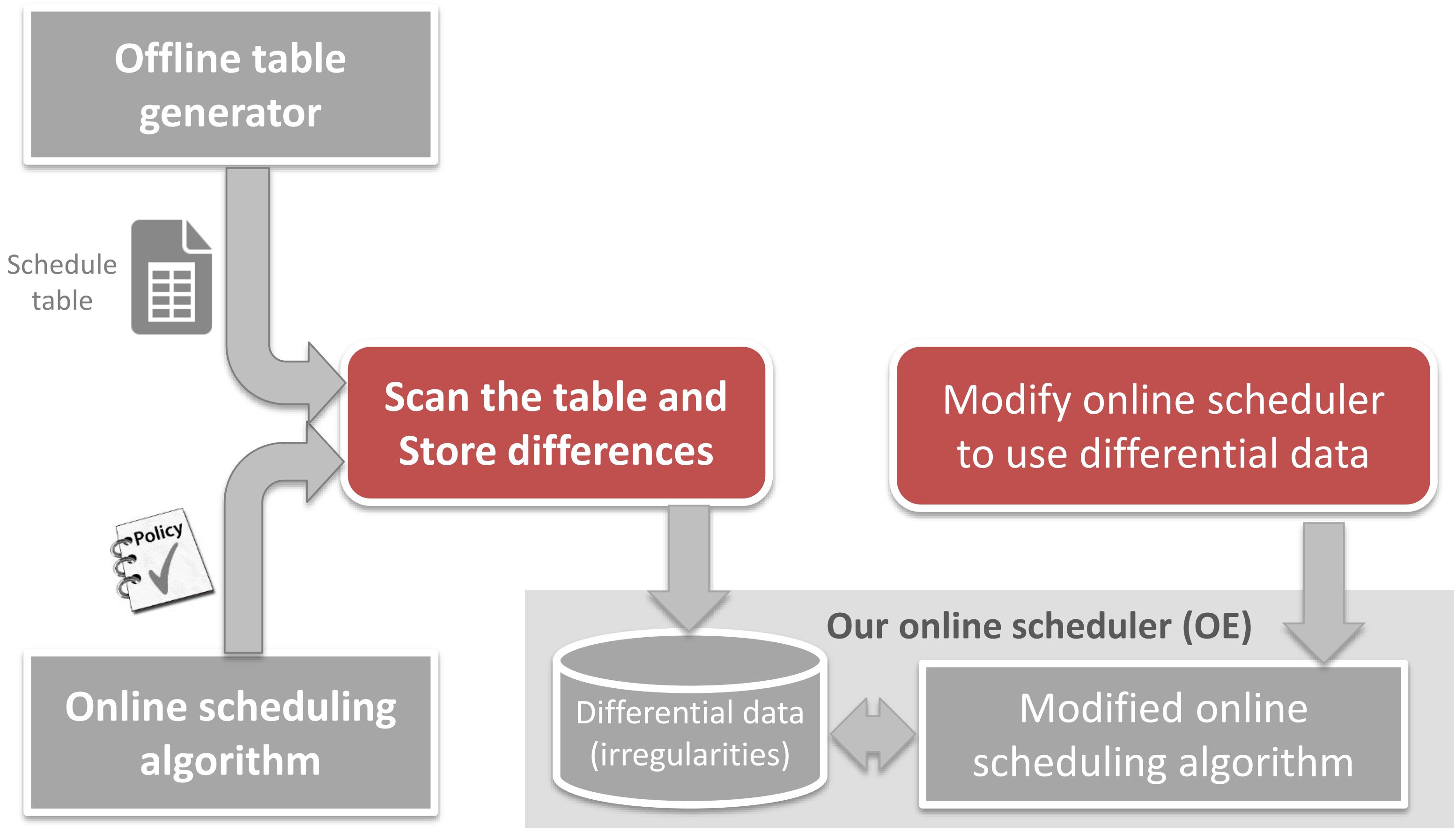
Agenda

- **Offline equivalence**



- Efficient table generation
- Evaluation
- Conclusion

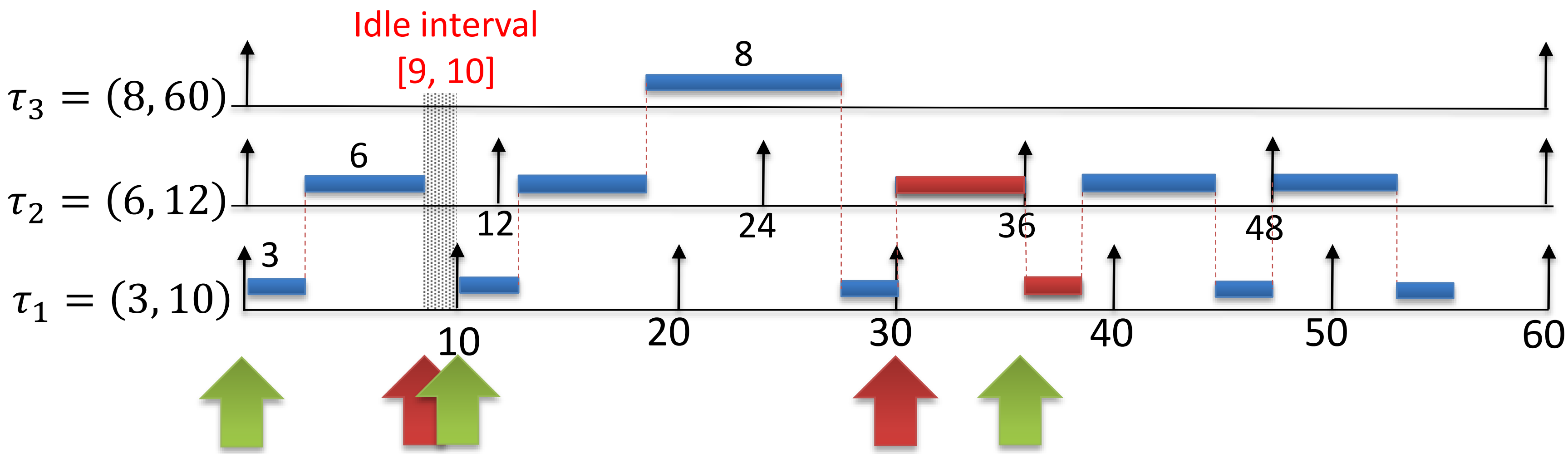
Two Key Components of Offline Equivalence



Scan Phase

- **Scan** the table to identify irregularities w.r.t. the online policy and **store** them
  - Priority inversion irregularity
  - Idle interval irregularity

Online policy:
rate monotonic



Only two entries were needed

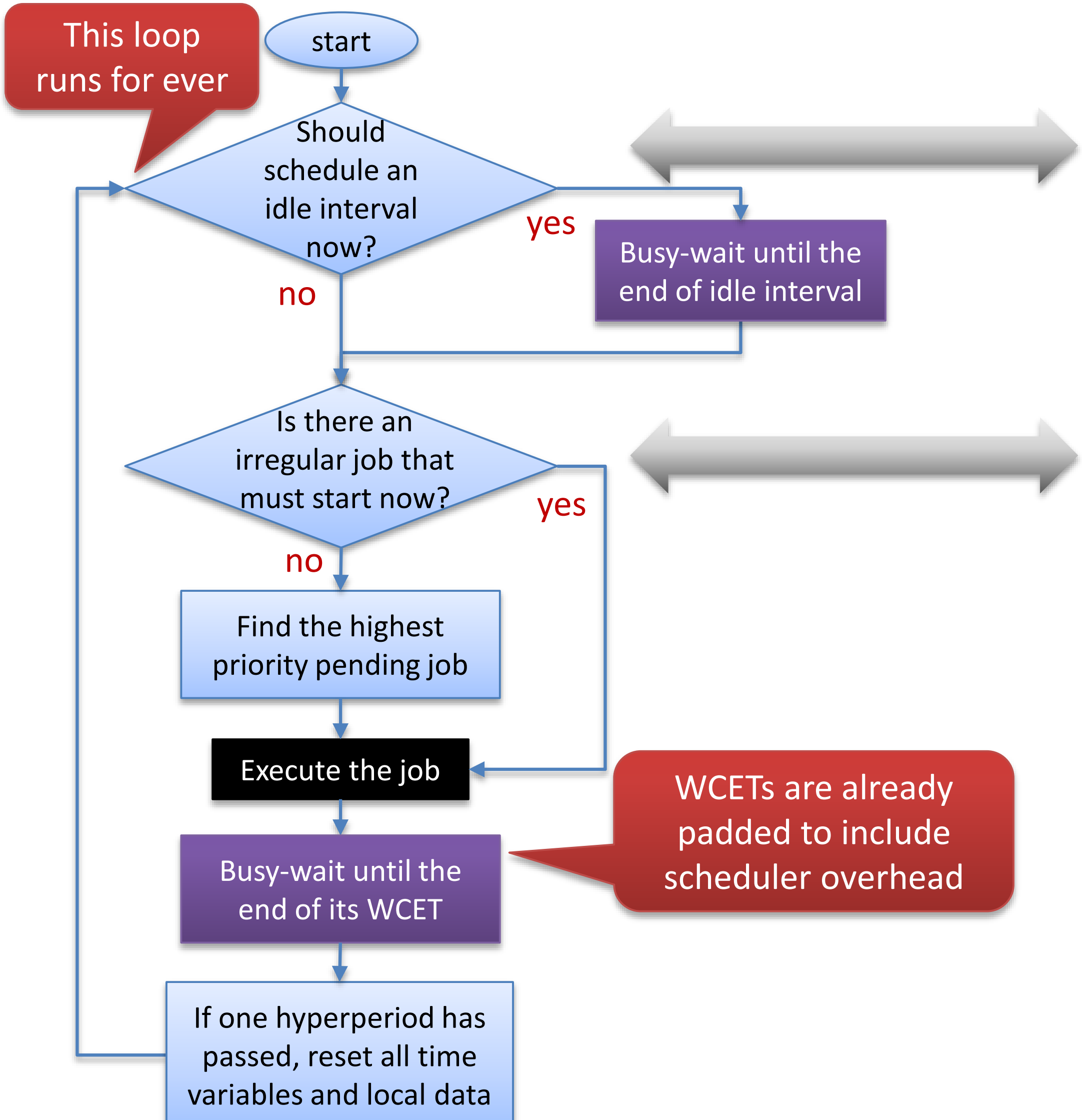
Idle-time irregularity table (IIT)

	From time 9, for 1 time unit

Priority inversion table (PIT)

	The 3rd Job of τ_2 starts at 30

Modifying Baseline Online Scheduler to Use Differential Data



Idle-time irregularity table (IIT)

(sorted by start time)

Start time	Duration

Priority inversion table (PIT)

(sorted by Task# and Job#)

Task #	Job #	Start time

Implementation

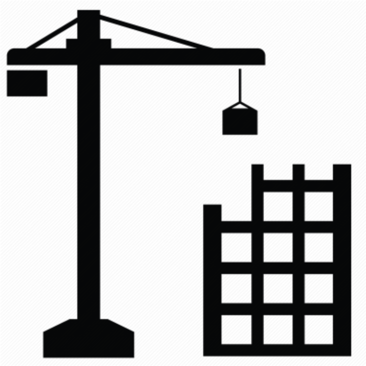
- **Baseline online scheduling policy**: non-preemptive RM

- **Implementation platform**: Arduino
  - Entire implementation of OE scheduler is just 200 lines of simple C++ code
  - Possibility to store extra tables:
    - in flash memory
    - in RAM

  - Available online at
    - People.mpi-sws.org/~bbb/papers/details/rtas17m/index.html

Agenda

- Offline equivalence approach
- **Efficient table generation**
- Evaluation
- Conclusion



Task model

- Periodic Tasks
- Constrained deadline
- No release offset

Strongly NP-Hard!

Why Non-preemptive Scheduling is Hard?

Branch and bound is a common approach [Moore68, Pinedo16, …]:

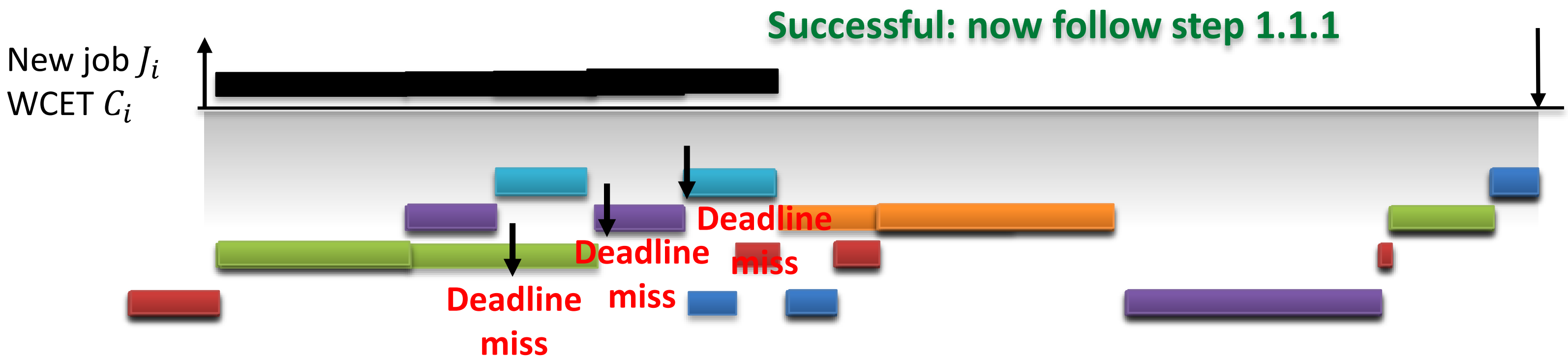
- Tries all possible combinations of the jobs in the ordering
- Even with pruning conditions it is still a combinatorial problem.

The original problem is ***job sequencing***:

- **Given** a set of jobs
- **Find** an ordering such that all timing constraints are met

A simpler approach:
iterative backtracking

1. **For each possible schedule** for $\boldsymbol{J_i}$
    1.1. If J_i and all other scheduled jobs meet their timing constrains
        1.1.1. **Recursively try to schedule** J_{i+1} (all other not scheduled jobs)
        1.1.2. If succeeded, return the schedule



This paper:

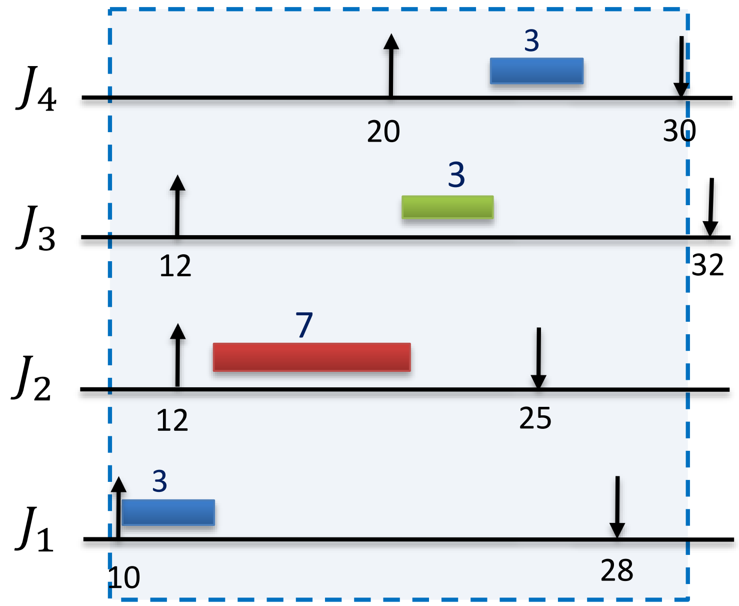
To reduce the backtracking steps and improve the search speed,

group jobs in *chained windows*!

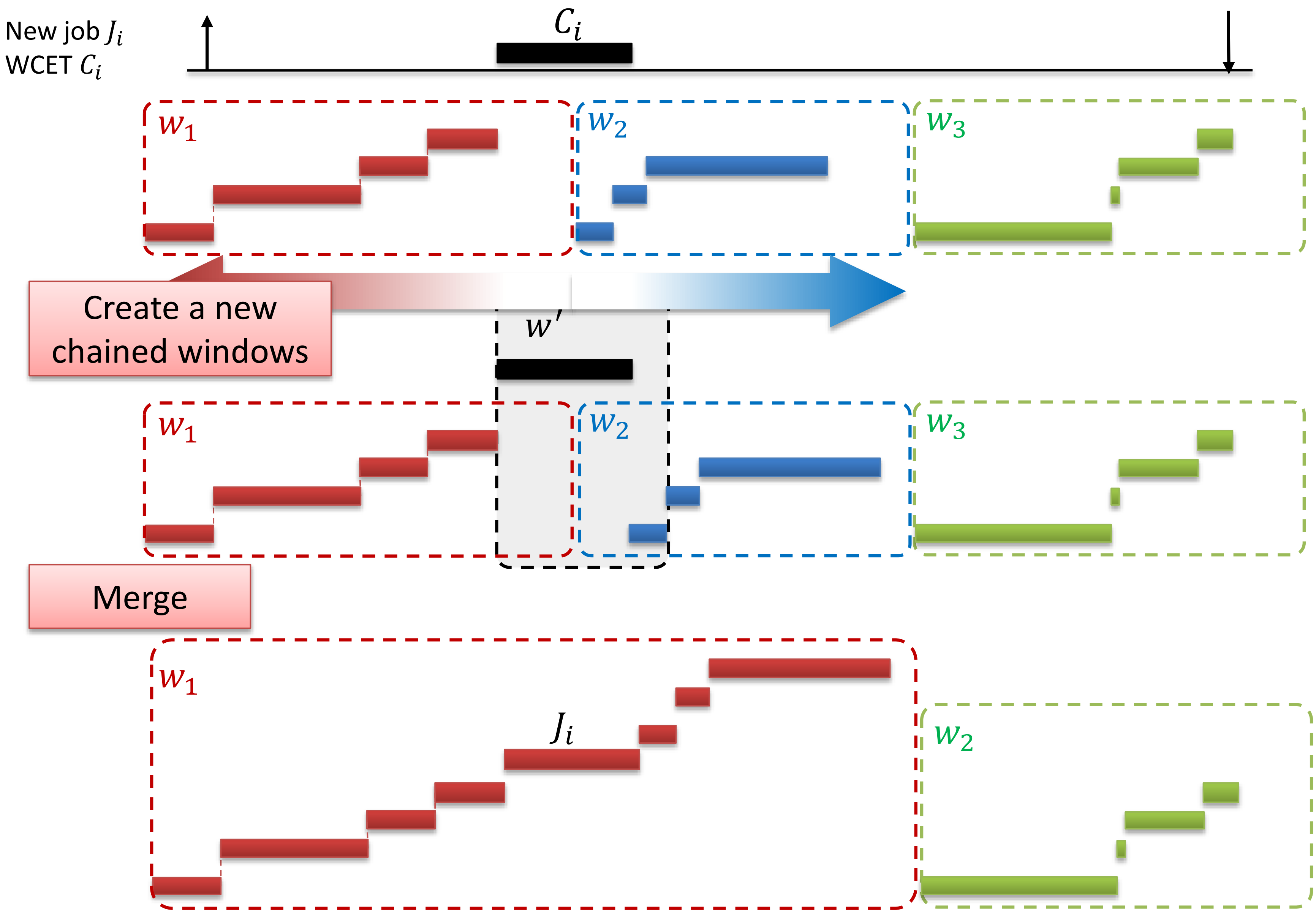
What is a Chained Window?

A **chained window** is a tuple that represents a job sequence, a window of time, and a slack value and

> any schedule that starts and finishes the **job sequence** within the **window**, respects all timing constraints of the jobs

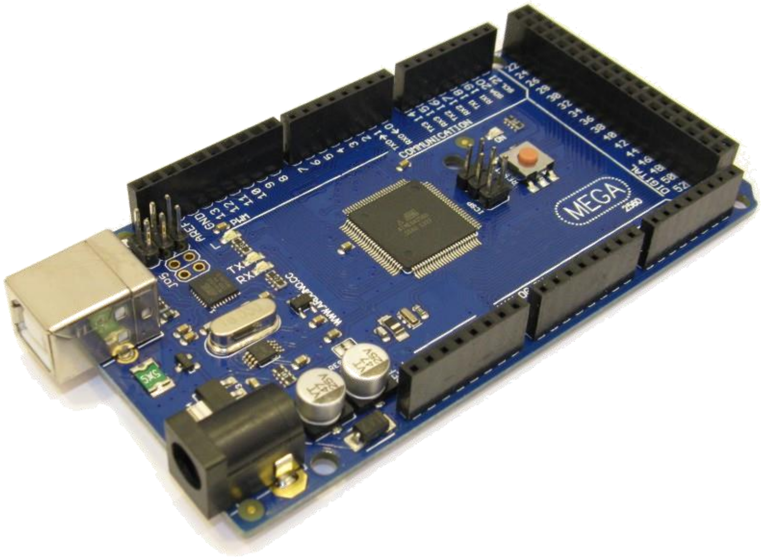


Chained Window Technique in a Nutshell



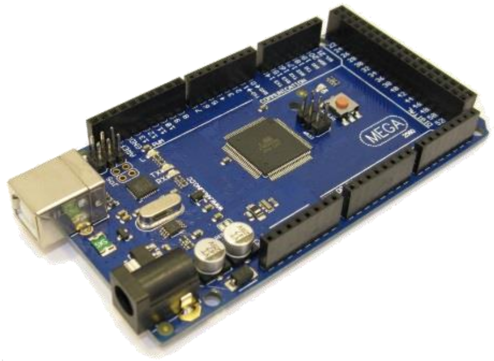
Agenda

- Offline equivalence approach
- Efficient offline table generation
- **Evaluation**
- Conclusion



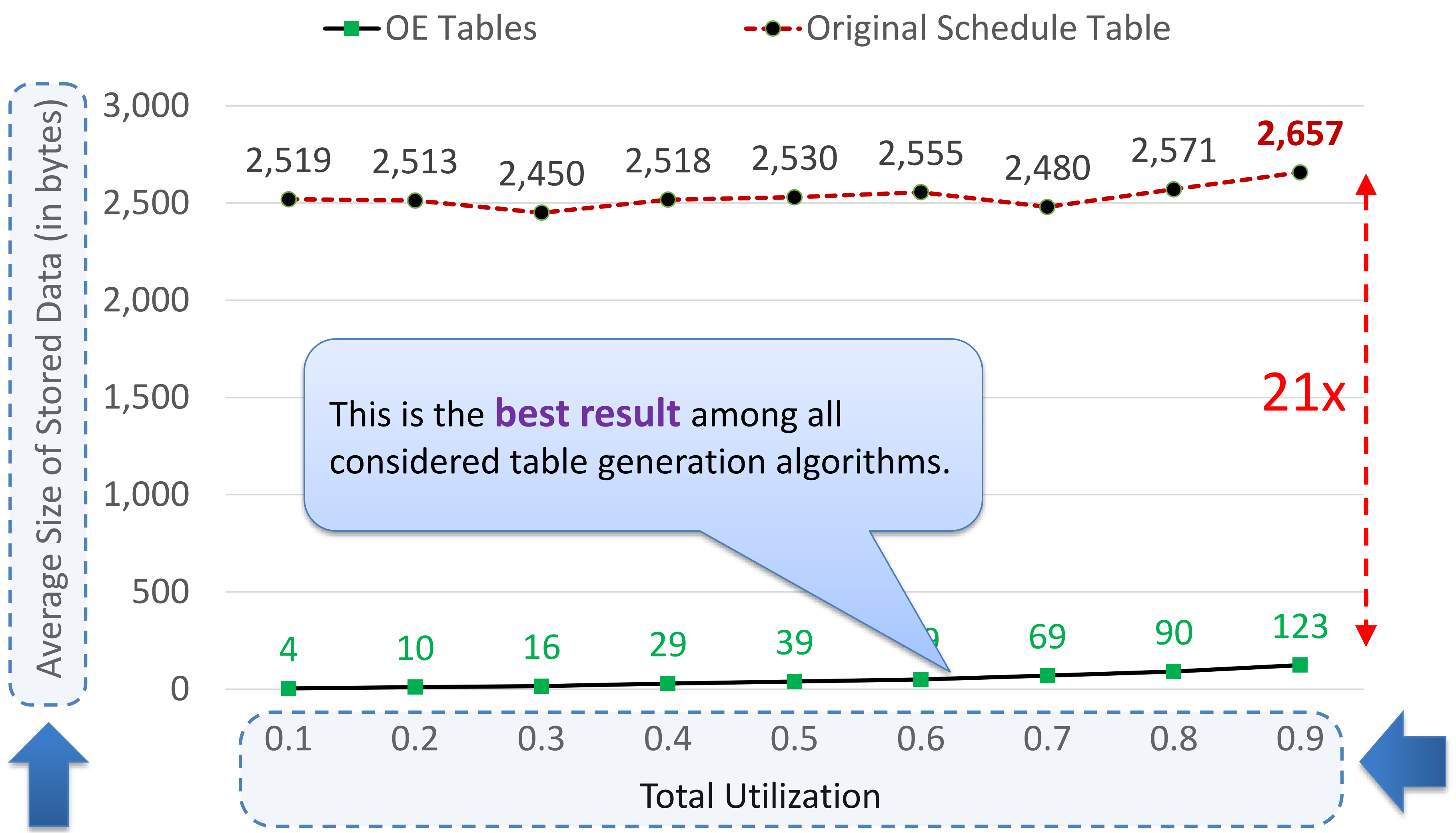
Main Questions

- **How efficient is Offline Equivalence (OE)?**
  - **What is the memory requirement of OE?**
  - **What is the timing overhead of OE online scheduler?**
  - Implementation platform:
    - Arduino Mega 5056
    - 6 KiB RAM, 256 KiB Flash memory, 16MHz processor speed
  - Measurements:
    - Required memory for OE tables (in Bytes)
    - OE online scheduler's run time (in microseconds)

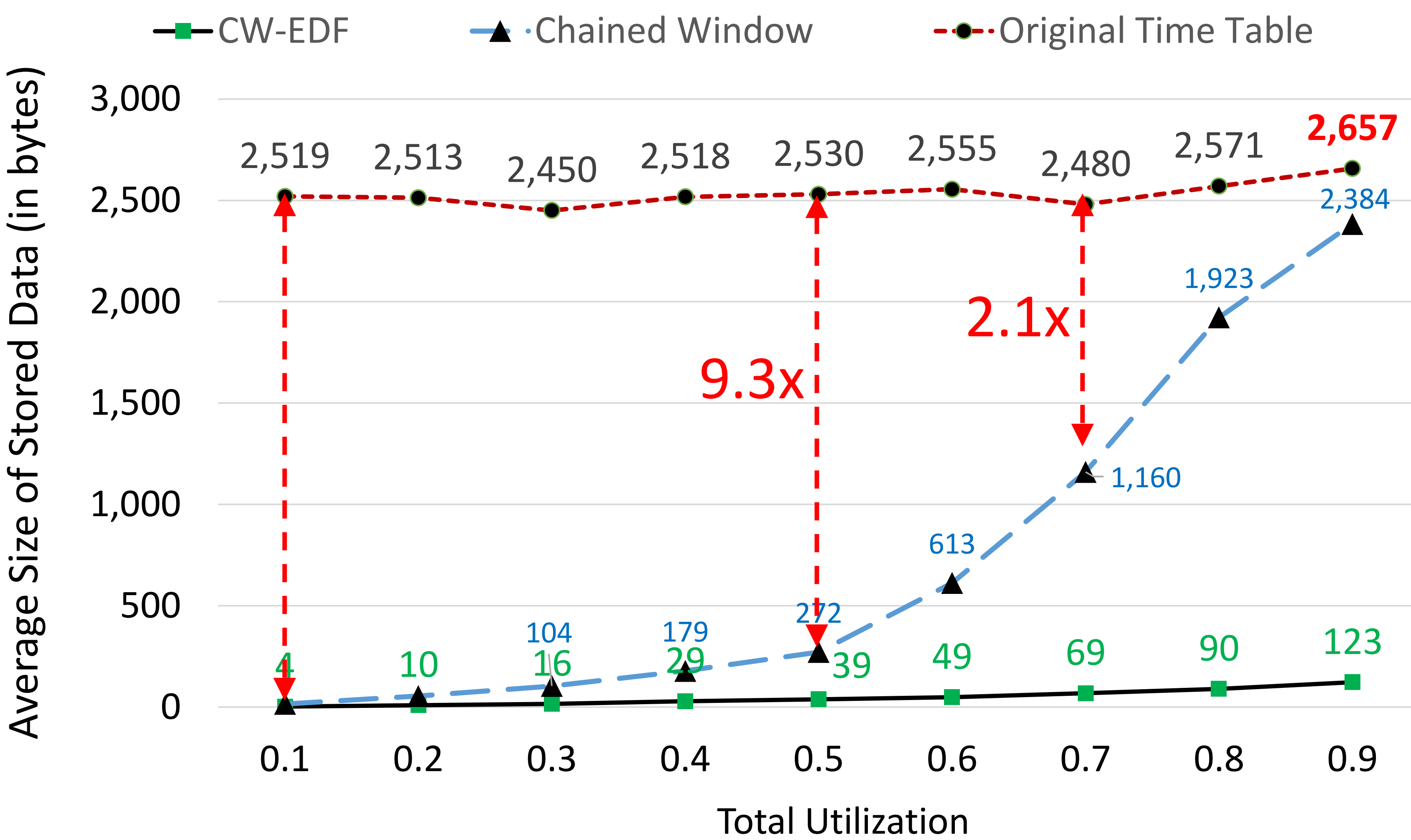


- **How fast and efficient is the Chained Window technique?**
  - Measurements:
    - Schedulability ratio for varying system utilization
    - Schedulability ratio for varying time budget

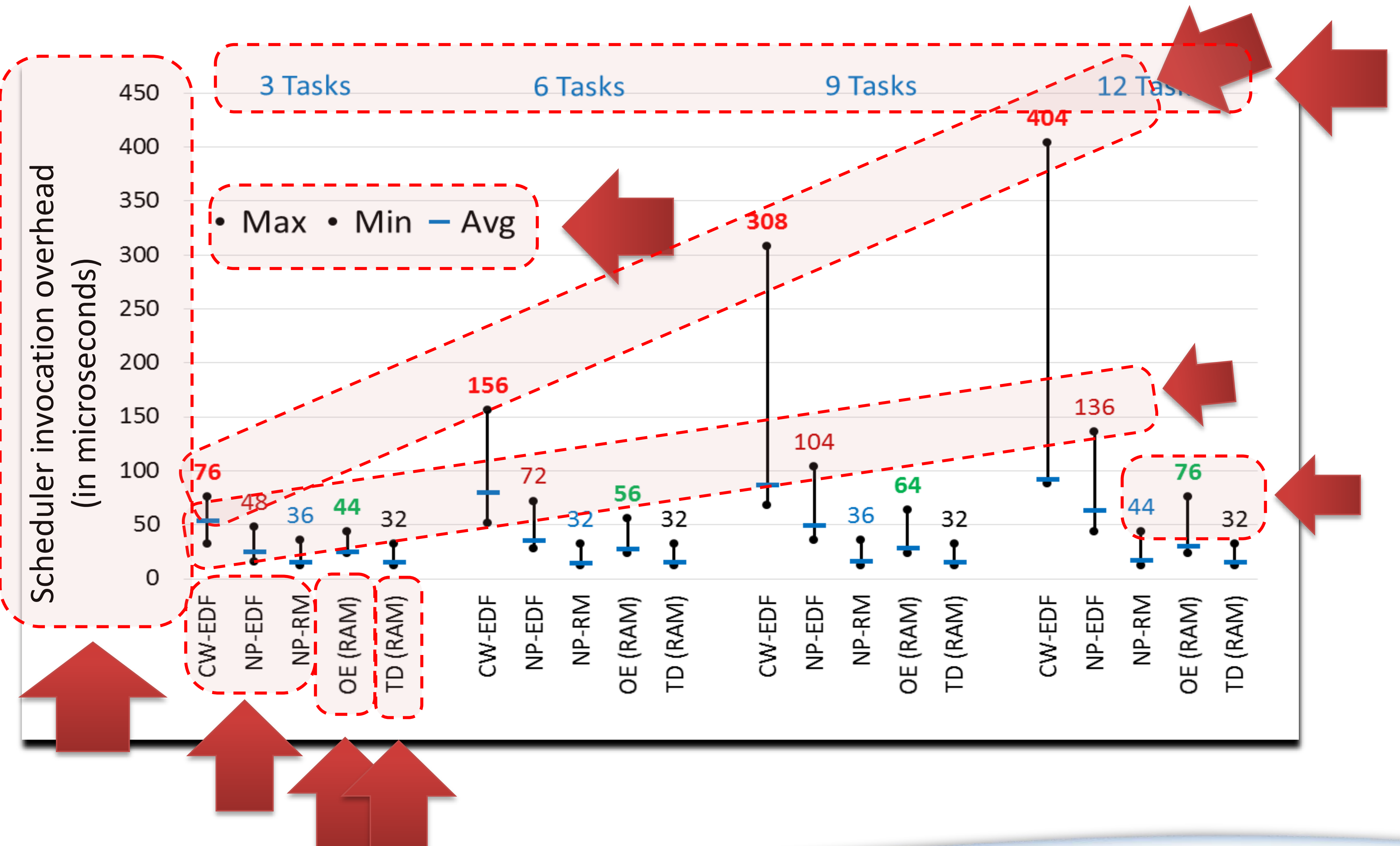
Offline Equivalence Reduces Memory Requirements



Memory Savings Depend on the Table Generation Algorithm



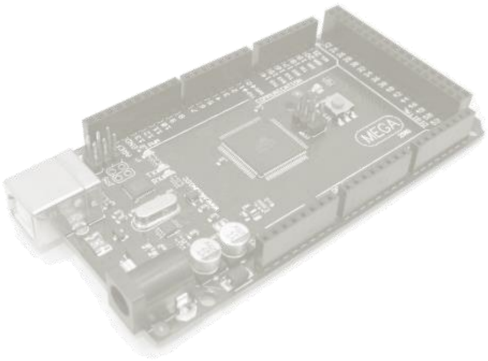
What is the Runtime Overhead of OE?



Main Questions

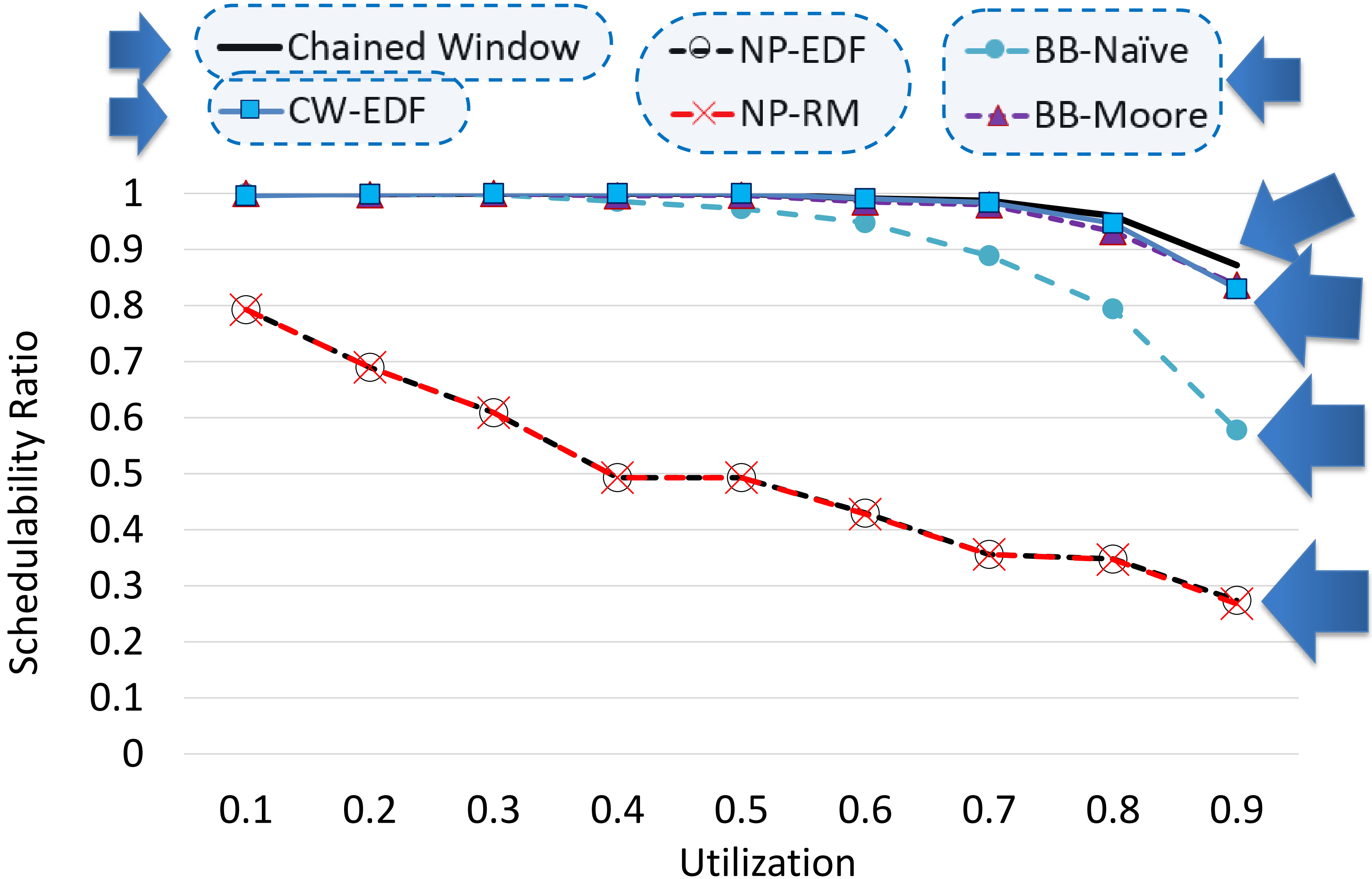
- How efficient is Offline Equivalence (OE)?
  - What is the memory requirement of OE?
  - What is the timing overhead of OE online scheduler?

  - Implementation platform:
    - Arduino Mega 5056
    - 6 KiB RAM, 256 KiB Flash memory, 16MHz processor speed
  - Measurements:
    - Required memory for OE tables (in Bytes)
    - OE online scheduler’s run time (in microseconds)

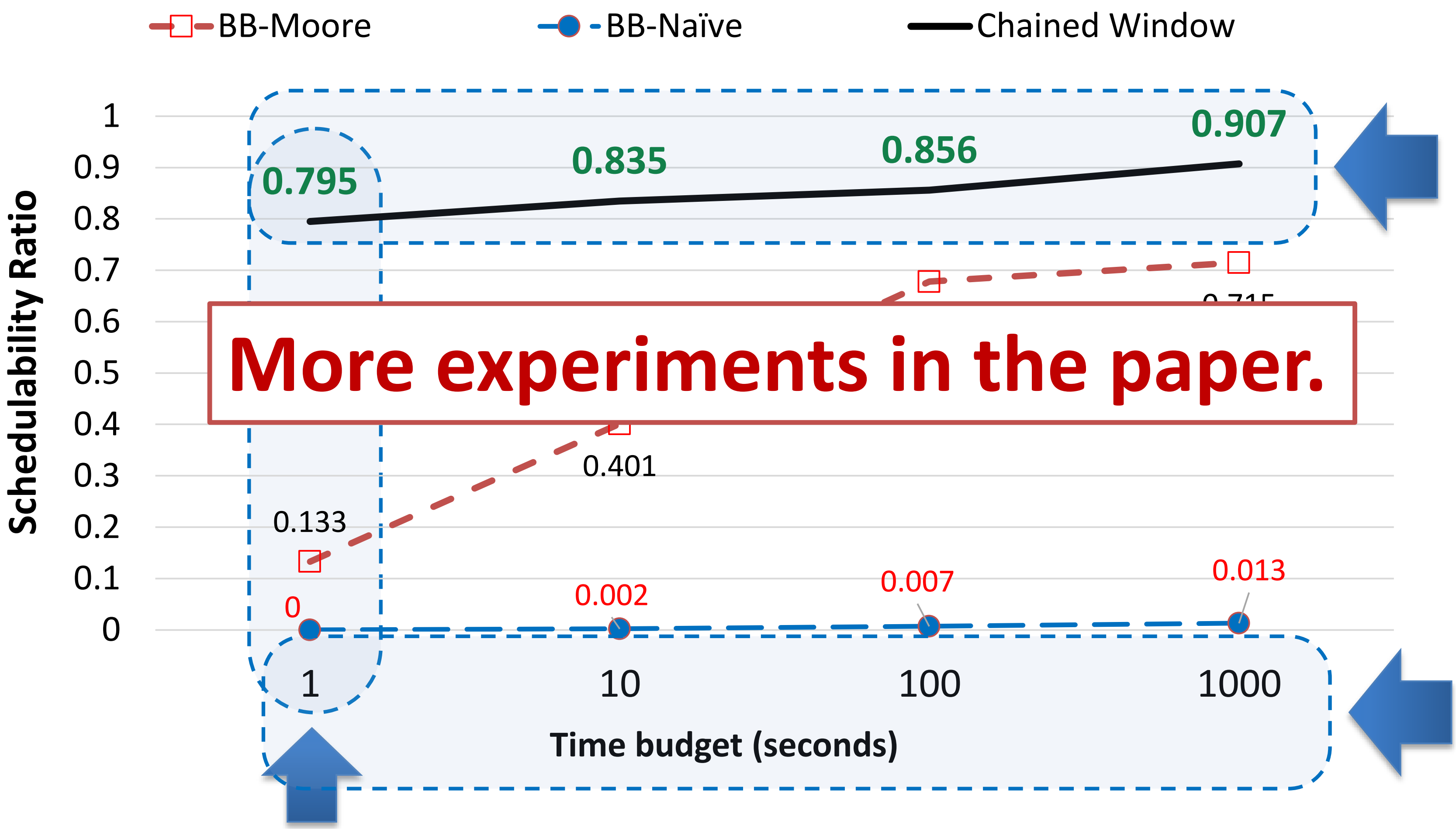


- **How fast and efficient is Chained Window technique?**
  - Measured outputs:
    - Schedulability ratio for varying system utilization
    - Schedulability ratio for varying time budget

How Efficient is the Chained Window Technique?



How Fast is the Chained Window Technique?



10 tasks per task set. Utilization 0.9.

Agenda

- Related work
- Offline equivalence approach
- Efficient offline table generation
- Evaluation
- **Conclusion and future work**

Summary and Conclusions

	What does it do?	What does it not do?
Offline Equivalence	Schedules task according to a given schedule Reduces memory consumption Has low runtime overhead	Guarantees that the extra required information fits in a the memory Minimizes memory consumption
Chained Window Technique	Is fast and efficient in generating a schedule	Optimal, i.e., is able to find a schedule for any feasible task set

Open Problems and Future Directions

Generate a schedule with the least number of irregularities

Find the **best policy, parameters** and **encoding** that minimizes the size of stored data

Find a set of differential **parameters** such that differential data fits in a given memory size

Max Planck Institute for Software Systems

Offline equivalence available at

http://people.mpi-sws.org/~bbb/papers/details/rtas17m/index.html

Thank you