Disclaimer

- The material provided in this document is not my original work and is a summary of some one else's work(s).
- A simple Google search of the title of the document will direct you to the original source of the material.
- I do not guarantee the accuracy, completeness, timeliness, validity, non-omission, merchantability or fitness of the contents of this document for any particular purpose.
- Downloaded from najeebkhan.github.io

FLEXRAY SCHEDULE OPTIMIZATION OF THE STATIC SEGMENT

Presented By
Najeeb
2013 – 5 – 16

OUTLINE

Problem Definition

Schedule Optimization

Results

Conclusions

PROBLEM DEFINITION

PROBLEM DEFINITION

- ✓ A schedule optimization that minimizes the number of used slots is necessary to allow a high flexibility for incremental schedule changes and for future automotive networks with a higher data volume

PROBLEM DEFINITION

- ✓ A schedule optimization that minimizes the number of used slots is necessary to allow a high flexibility for incremental schedule changes and for future automotive networks with a higher data volume
- ✓ Cycle multiplexing is used to increase the utilization of the FlexRay bus

PROBLEM DEFINITION

- ✓ A schedule optimization that minimizes the number of used slots is necessary to allow a high flexibility for incremental schedule changes and for future automotive networks with a higher data volume
- ✓ Cycle multiplexing is used to increase the utilization of the FlexRay bus
- ✓ The cycle multiplexing of PDUs is defined by the base cycle and the cycle repetition

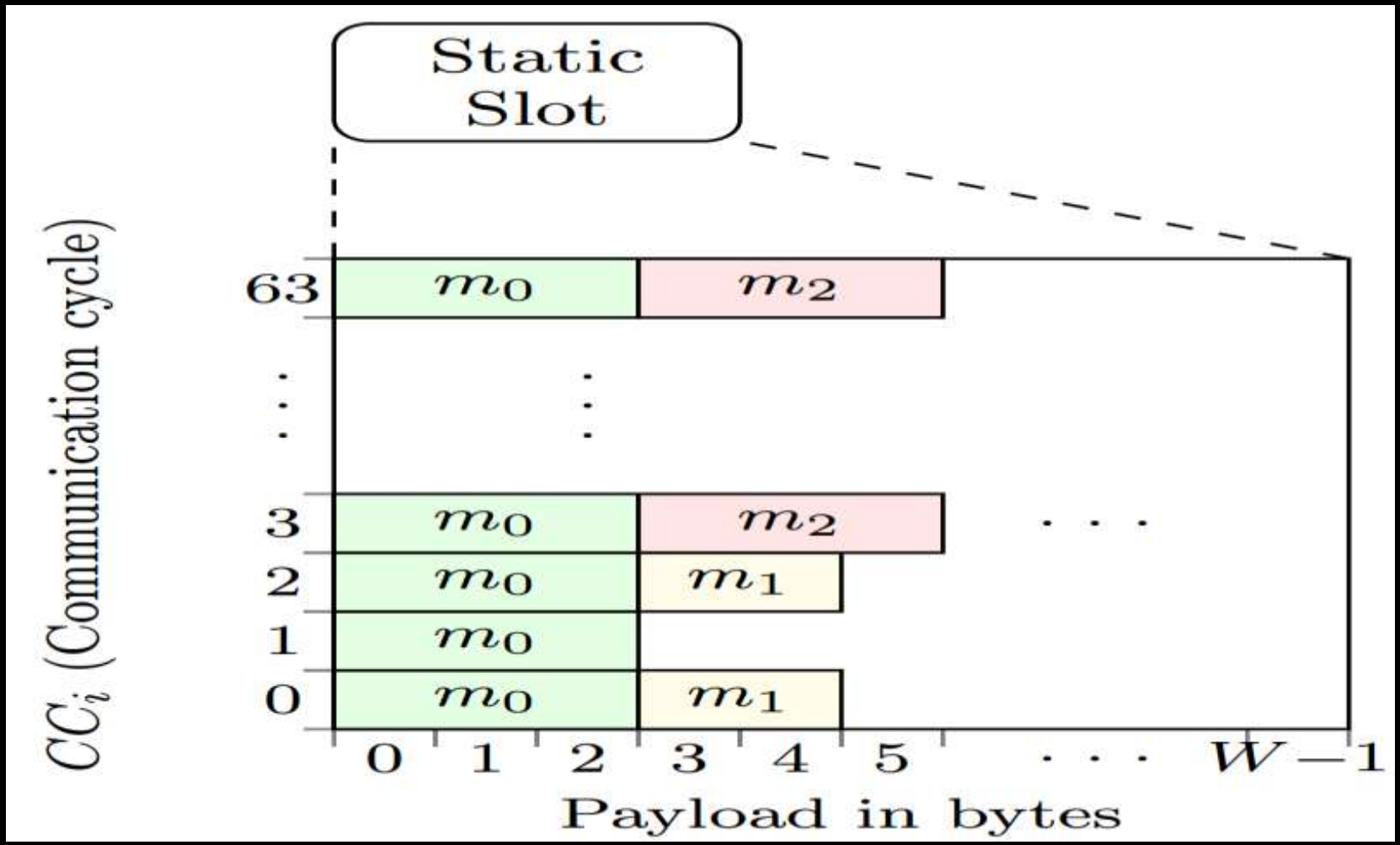
PROBLEM DEFINITION

- ✓ A schedule optimization that minimizes the number of used slots is necessary to allow a high flexibility for incremental schedule changes and for future automotive networks with a higher data volume
- ✓ Cycle multiplexing is used to increase the utilization of the FlexRay bus
- ✓ The cycle multiplexing of PDUs is defined by the base cycle and the cycle repetition
- ✓ The base cycle defines the offset in cycles for the first occurrence of the respective PDU

PROBLEM DEFINITION

- ✓ A schedule optimization that minimizes the number of used slots is necessary to allow a high flexibility for incremental schedule changes and for future automotive networks with a higher data volume
- ✓ Cycle multiplexing is used to increase the utilization of the FlexRay bus
- ✓ The cycle multiplexing of PDUs is defined by the base cycle and the cycle repetition
- ✓ The base cycle defines the offset in cycles for the first occurrence of the respective PDU
- ✓ The cycle repetition denotes the frequency of a PDU in the multiplexing and is always a power of two 2^n

PROBLEM DEFINITION



PROBLEM DEFINITION

PROBLEM DEFINITION

- For a given base cycle b and repetition r, a PDU is existent in each communication cycle CC_i with

PROBLEM DEFINITION

- For a given base cycle b and repetition r, a PDU is existent in each communication cycle CC_i with

$$i = (b + r \cdot n)\%64 \text{ with } n \in \mathbb{N}_0.$$

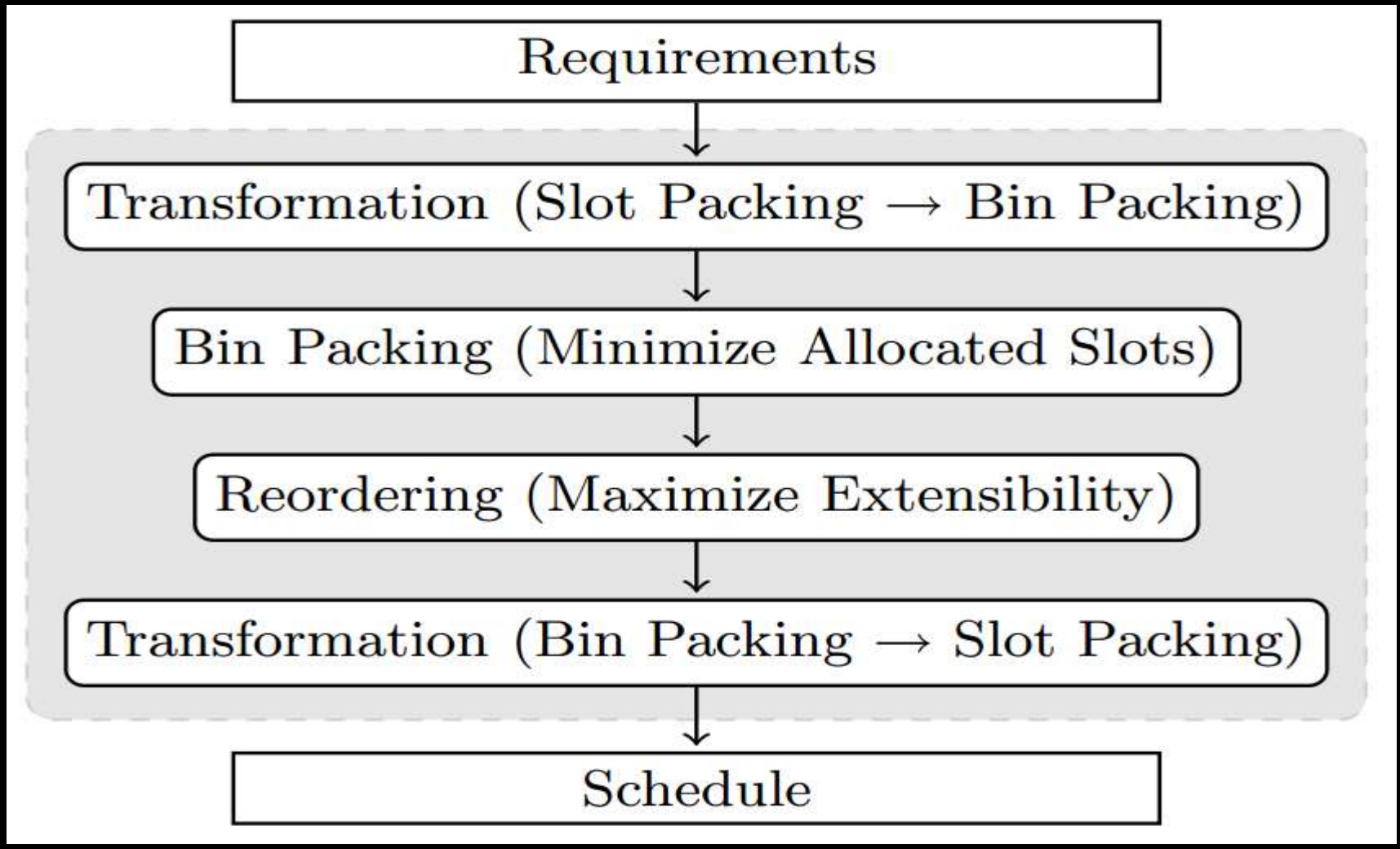
PROBLEM DEFINITION

- For a given base cycle b and repetition r, a PDU is existent in each communication cycle CC_i with

$$i = (b + r \cdot n)\%64 \text{ with } n \in \mathbb{N}_0.$$

- The goal is to minimize the number of slots to maximize the utilization.

SCHEDULE OPTIMIZATION



PROBLEM TRANSFORMATION

PROBLEM TRANSFORMATION

- ✓ A one-to-one transformation between the slot packing problem that arises from the FlexRay cycle multiplexing and a special form of a two dimensional bin packing problem

PROBLEM TRANSFORMATION

- ✓ A one-to-one transformation between the slot packing problem that arises from the FlexRay cycle multiplexing and a special form of a two dimensional bin packing problem
- ✓ Each slot corresponds to one bin

PROBLEM TRANSFORMATION

- ✓ A one-to-one transformation between the slot packing problem that arises from the FlexRay cycle multiplexing and a special form of a two dimensional bin packing problem
- ✓ Each slot corresponds to one bin
- ✓ Each slot is defined by the payload size W and the number of cycles H which is 64 for the FlexRay bus

PROBLEM TRANSFORMATION

- ✓ A one-to-one transformation between the slot packing problem that arises from the FlexRay cycle multiplexing and a special form of a two dimensional bin packing problem
- ✓ Each slot corresponds to one bin
- ✓ Each slot is defined by the payload size W and the number of cycles H which is 64 for the FlexRay bus
- ✓ The set of PDUs is denoted M

PROBLEM TRANSFORMATION

- ✓ A one-to-one transformation between the slot packing problem that arises from the FlexRay cycle multiplexing and a special form of a two dimensional bin packing problem
- ✓ Each slot corresponds to one bin
- ✓ Each slot is defined by the payload size W and the number of cycles H which is 64 for the FlexRay bus
- ✓ The set of PDUs is denoted M
- ✓ Each PDU m is defined by

PROBLEM TRANSFORMATION

- ✓ A one-to-one transformation between the slot packing problem that arises from the FlexRay cycle multiplexing and a special form of a two dimensional bin packing problem
- ✓ Each slot corresponds to one bin
- ✓ Each slot is defined by the payload size W and the number of cycles H which is 64 for the FlexRay bus
- ✓ The set of PDUs is denoted M
- ✓ Each PDU m is defined by
- ✓ w_m : length in bytes

PROBLEM TRANSFORMATION

- ✓ A one-to-one transformation between the slot packing problem that arises from the FlexRay cycle multiplexing and a special form of a two dimensional bin packing problem
- ✓ Each slot corresponds to one bin
- ✓ Each slot is defined by the payload size W and the number of cycles H which is 64 for the FlexRay bus
- ✓ The set of PDUs is denoted M
- ✓ Each PDU m is defined by
- ✓ w_m : length in bytes
- ✓ r_m : repetitions in the powers of two

PROBLEM TRANSFORMATION

- ✓ For a feasible slot packing, two values for each PDU have to be determined
- ✓ x_m : Offset in bytes on the x-axis
- ✓ b_m : The base cycle that defines the offset on the y-axis. It holds that $b_m < r_m$
- ✓ The task of the transformation of the slot packing into a bin packing is to convert each PDU into a rectangle element and determine its position such that each feasible slot packing results in a feasible bin packing and vice versa
- ✓ The bin size is the same as the slot size with width W and height H

PROBLEM TRANSFORMATION

PROBLEM TRANSFORMATION

- ✓ The position on the x-axis x_m and the width w_m for each element m correspond to the position and width of the PDU in the slot packing

PROBLEM TRANSFORMATION

- ✓ The position on the x-axis x_m and the width w_m for each element m correspond to the position and width of the PDU in the slot packing
- ✓ The main task is to find a transformation that obtains the following two values for each element

PROBLEM TRANSFORMATION

- The position on the x-axis x_m and the width w_m for each element m correspond to the position and width of the PDU in the slot packing
- The main task is to find a transformation that obtains the following two values for each element
- y_m: The offset on the y-axis

PROBLEM TRANSFORMATION

- ✓ The position on the x-axis x_m and the width w_m for each element m correspond to the position and width of the PDU in the slot packing
- ✓ The main task is to find a transformation that obtains the following two values for each element
- ✓ y_m: The offset on the y-axis
- ✓ h_m: The height of an element

PROBLEM TRANSFORMATION

- ✓ The position on the x-axis x_m and the width w_m for each element m correspond to the position and width of the PDU in the slot packing
- ✓ The main task is to find a transformation that obtains the following two values for each element
- ✓ y_m: The offset on the y-axis
- ✓ h_m: The height of an element
- ✓ The transformation for the height h_m is related to the repetition r_m

PROBLEM TRANSFORMATION

- ✓ The position on the x-axis x_m and the width w_m for each element m correspond to the position and width of the PDU in the slot packing
- ✓ The main task is to find a transformation that obtains the following two values for each element
- ✓ y_m: The offset on the y-axis
- ✓ h_m: The height of an element
- ✓ The transformation for the height h_m is related to the repetition r_m

$$h_m = \frac{H}{r_m}$$

$$r_m = \frac{H}{h_m}$$

PROBLEM TRANSFORMATION

PROBLEM TRANSFORMATION

- ✓ The level of an element in the bin is $l_m = y_m/h_m$

PROBLEM TRANSFORMATION

- ✓ The level of an element in the bin is $l_m = y_m/h_m$
- ✓ A transformation function t is defined which directly transforms the level of an element l_m to base b_m and vice versa such that the following holds

PROBLEM TRANSFORMATION

- ✓ The level of an element in the bin is $l_m = y_m/h_m$
- ✓ A transformation function t is defined which directly transforms the level of an element l_m to base b_m and vice versa such that the following holds

$$l_m = t(b_m, r_m) \text{ and } b_m = t(l_m, r_m)$$

$$y_m = l_m \cdot h_m = t(b_m, r_m) \cdot \frac{H}{r_m}$$

$$b_m = t(\frac{y_m}{h_m}, r_m) = t(\frac{y_m}{h_m}, \frac{H}{h_m}).$$

PROBLEM TRANSFORMATION

PROBLEM TRANSFORMATION

Transformation to Bin Problem:

$$l_m = t(b_m, r_m) \text{ and } b_m = t(l_m, r_m)$$

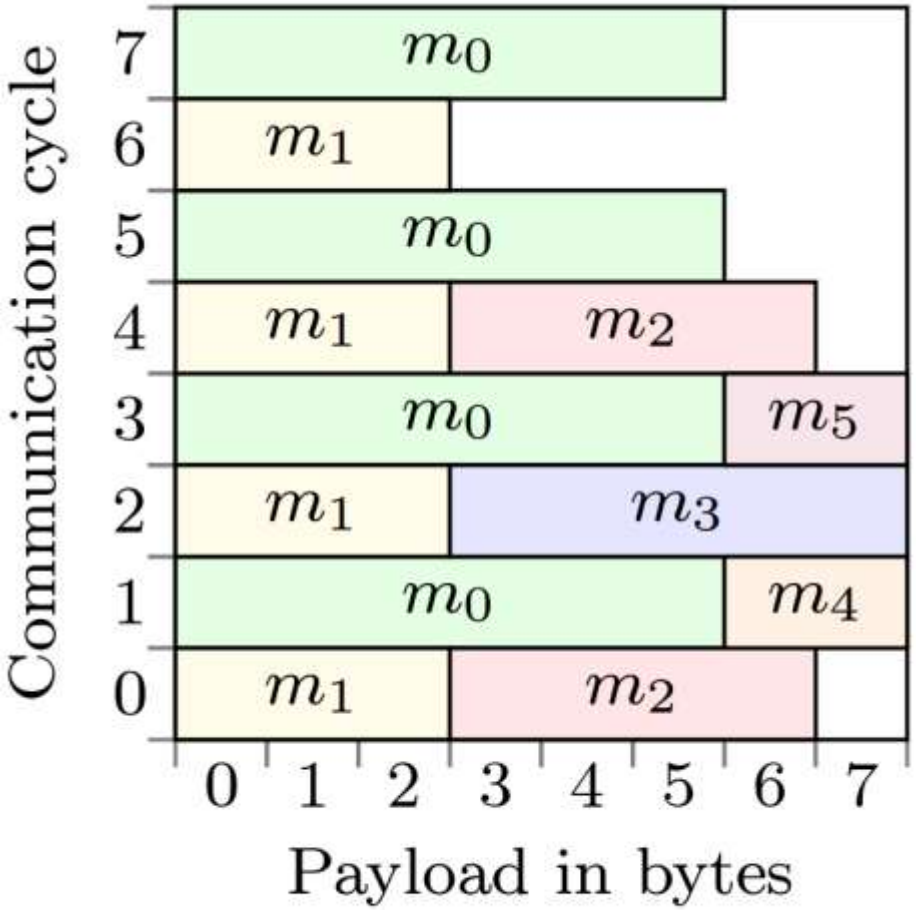
$$y_m = l_m \cdot h_m = t(b_m, r_m) \cdot \frac{H}{r_m}$$

Transformation to Slot Problem:

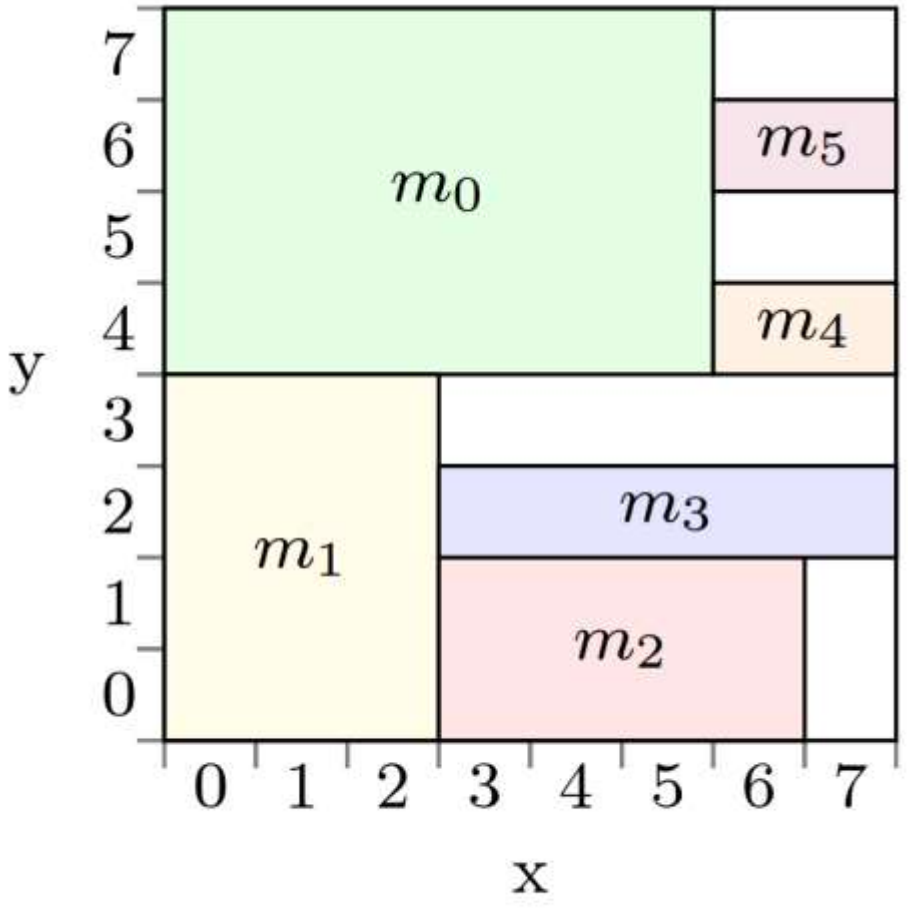
$$b_m = t(\frac{y_m}{h_m}, r_m) = t(\frac{y_m}{h_m}, \frac{H}{h_m}).$$

$$h_m = \frac{H}{r_m}$$

PROBLEM TRANSFORMATION



(a) Slot



(b) Bin

BIN PACKING

BIN PACKING

- The task of a two-dimensional bin packing problem is to pack rectangular elements of different sizes defined by an individual width w and height h into a minimal number of fixed size rectangular bins without any intersection

BIN PACKING

- ✓ The task of a two-dimensional bin packing problem is to pack rectangular elements of different sizes defined by an individual width w and height h into a minimal number of fixed size rectangular bins without any intersection
- ✓ The transformed problem has two constraints

BIN PACKING

- ✓ The task of a two-dimensional bin packing problem is to pack rectangular elements of different sizes defined by an individual width w and height h into a minimal number of fixed size rectangular bins without any intersection
- ✓ The transformed problem has two constraints
- ✓ Each element m has a height h_m that is a power of two

BIN PACKING

- ✓ The task of a two-dimensional bin packing problem is to pack rectangular elements of different sizes defined by an individual width w and height h into a minimal number of fixed size rectangular bins without any intersection
- ✓ The transformed problem has two constraints
- ✓ Each element m has a height h_m that is a power of two
- ✓ Each element m can be placed everywhere on the x-axis but only on a multiple of its height on the y-axis, i.e., $y_m = l \cdot h_m$

FAST GREEDY HEURISTIC

FAST GREEDY HEURISTIC

- ✓ The presented bin packing problem can be solved by a fast greedy heuristic

FAST GREEDY HEURISTIC

- ✓ The presented bin packing problem can be solved by a fast greedy heuristic

Algorithm 1 Fast greedy heuristic for bin packing.

```
1: S = {} //set of bins
2: for m ∈ M do
3:     for s ∈ S do
4:         if place(m, s) then
5:             continue with next m
6:         end if
7:     end for
8:     create new s and add it to S
9:     place(m, s)
10: end for
```

INTEGER LINEAR PROGRAMMING

INTEGER LINEAR PROGRAMMING

- ✓ The ILP formulation relies on the following binary variables

INTEGER LINEAR PROGRAMMING

- ✓ The ILP formulation relies on the following binary variables
- ✓ $m_{s,l}$: element m is placed at level l in bin s

INTEGER LINEAR PROGRAMMING

- ✓ The ILP formulation relies on the following binary variables
- ✓ $m_{s,l}$: element m is placed at level l in bin s
- ✓ s – bin s is allocated

INTEGER LINEAR PROGRAMMING

- ✓ The ILP formulation relies on the following binary variables
- ✓ $m_{s,l}$: element m is placed at level l in bin s
- ✓ s – bin s is allocated
- ✓ The ILP is formulated as follows

INTEGER LINEAR PROGRAMMING

$$min \sum_{s \in S} \mathbf{s}$$

$$\forall\ m \in M : \sum_{s \in S} \sum_{l=0}^{\frac{H}{h_m}-1} \mathbf{m_{s,l}} = 1$$

$$\forall\ s \in S, \{y = 0, ..., H-1\} : \sum_{m \in M} w_m \cdot \mathbf{m}_{\mathbf{s},\left\lfloor \frac{\mathbf{y}}{\mathbf{h_m}} \right\rfloor} \leq W$$

$$\forall\ m \in M, s \in S, \{l = 0, ..., \frac{H}{h_m} - 1\} : \mathbf{s} - \mathbf{m_{s,l}} \geq 0$$

INTEGER LINEAR PROGRAMMING

INTEGER LINEAR PROGRAMMING

- ✓ Solving the ILP provides a bin s and level l for each element m

INTEGER LINEAR PROGRAMMING

- ✓ Solving the ILP provides a bin s and level l for each element m
- ✓ The complexity of an ILP is exponential in general

INTEGER LINEAR PROGRAMMING

- ✓ Solving the ILP provides a bin s and level l for each element m
- ✓ The complexity of an ILP is exponential in general
- ✓ In contrast to the heuristic approach, the presented ILP cannot be used incrementally, i.e., an already allocated bin cannot be filled with additional elements without moving the old elements

ENHANCED ILP

$$lb(M) = \frac{\sum_{m \in M} w_m \cdot h_m}{W \cdot H}$$

$$\sum_{s \in S} \mathbf{s} \geq \lceil lb(M) \rceil$$

ENHANCED ILP

- ✓ The stated ILP can be further improved by reducing the search space by applying domain-specific knowledge

$$lb(M) = \frac{\sum_{m \in M} w_m \cdot h_m}{W \cdot H}$$

$$\sum_{s \in S} \mathbf{s} \geq \lceil lb(M) \rceil$$

ENHANCED ILP

- ✓ The stated ILP can be further improved by reducing the search space by applying domain-specific knowledge
- ✓ The set of S is reduced by one bin, simplifying the ILP by omitting several variables and constraints

$$lb(M) = \frac{\sum_{m \in M} w_m \cdot h_m}{W \cdot H}$$

$$\sum_{s \in S} \mathbf{s} \geq \lceil lb(M) \rceil$$

ENHANCED ILP

- ✓ The stated ILP can be further improved by reducing the search space by applying domain-specific knowledge
- ✓ The set of S is reduced by one bin, simplifying the ILP by omitting several variables and constraints
- ✓ In case there exists no feasible solution of this simplified ILP the reference solution obtained by the heuristic is already optimal

$$lb(M) = \frac{\sum_{m \in M} w_m \cdot h_m}{W \cdot H}$$

$$\sum_{s \in S} \mathbf{s} \geq \lceil lb(M) \rceil$$

ENHANCED ILP

- ✓ The stated ILP can be further improved by reducing the search space by applying domain-specific knowledge
- ✓ The set of S is reduced by one bin, simplifying the ILP by omitting several variables and constraints
- ✓ In case there exists no feasible solution of this simplified ILP the reference solution obtained by the heuristic is already optimal
- ✓ A lower bound for the objective is deduced by domain-specific knowledge to improve the runtime of the ILP

$$lb(M) = \frac{\sum_{m \in M} w_m \cdot h_m}{W \cdot H}$$

$$\sum_{s \in S} \mathbf{s} \geq \lceil lb(M) \rceil$$

MUTEX PACKING

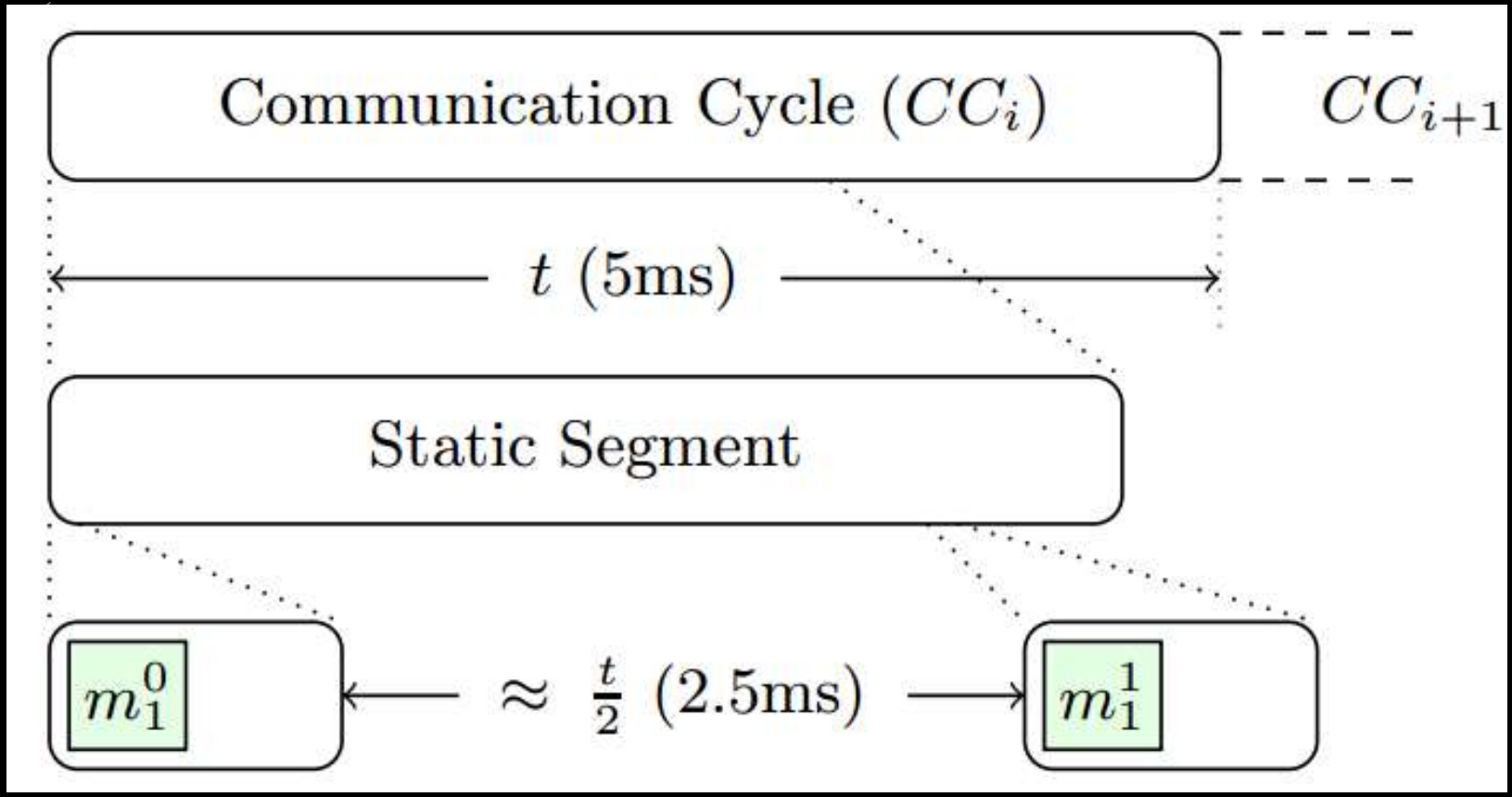
MUTEX PACKING

- In some cases, it becomes necessary to add mutual exclusion (mutex) to the bin packing, i.e., two elements are not allowed to be packed into the same bin

MUTEX PACKING

- ✓ In some cases, it becomes necessary to add mutual exclusion (mutex) to the bin packing, i.e., two elements are not allowed to be packed into the same bin
- ✓ This requirement arises if the scheduling problem uses the in-cycle repetition of PDUs

MUTEX PACKING



REORDERING FOR EXTENSIBILITY

$$U(s) = \sum_{m \in s} \frac{w_m}{W} \cdot \frac{h_m}{H}$$

REORDERING FOR EXTENSIBILITY

- ✓ The void space in partially used slots is exploited to schedule new PDUs

$$U(s) = \sum_{m \in s} \frac{w_m}{W} \cdot \frac{h_m}{H}$$

REORDERING FOR EXTENSIBILITY

- ✓ The void space in partially used slots is exploited to schedule new PDUs
- ✓ The extensibility of a slot describes its capability to allow scheduling additional PDUs

$$U(s) = \sum_{m \in s} \frac{w_m}{W} \cdot \frac{h_m}{H}$$

REORDERING FOR EXTENSIBILITY

- ✓ The void space in partially used slots is exploited to schedule new PDUs
- ✓ The extensibility of a slot describes its capability to allow scheduling additional PDUs
- ✓ The utilization of a bin is defined as

$$U(s) = \sum_{m \in s} \frac{w_m}{W} \cdot \frac{h_m}{H}$$

REORDERING FOR EXTENSIBILITY

- ✓ The void space in partially used slots is exploited to schedule new PDUs
- ✓ The extensibility of a slot describes its capability to allow scheduling additional PDUs
- ✓ The utilization of a bin is defined as

$$U(s) = \sum_{m \in s} \frac{w_m}{W} \cdot \frac{h_m}{H}$$

- ✓ In general, a lower utilization leads to a higher extensibility of a bin

REORDERING FOR EXTENSIBILITY

- ✓ The void space in partially used slots is exploited to schedule new PDUs
- ✓ The extensibility of a slot describes its capability to allow scheduling additional PDUs
- ✓ The utilization of a bin is defined as

$$U(s) = \sum_{m \in s} \frac{w_m}{W} \cdot \frac{h_m}{H}$$

- ✓ In general, a lower utilization leads to a higher extensibility of a bin
- ✓ The void space of a bin should be connected to enable the placement of large elements as well as several small elements

REORDERING FOR EXTENSIBILITY

$$E(s) = 1 - U(s) - \frac{w_{max}}{W} \cdot \frac{h_{max}}{H}$$

REORDERING FOR EXTENSIBILITY

- ✓ The goal is to re-order the elements in a bin in order to maximize a maximal rectangle over the void space

$$E(s) = 1 - U(s) - \frac{w_{max}}{W} \cdot \frac{h_{max}}{H}$$

REORDERING FOR EXTENSIBILITY

- ✓ The goal is to re-order the elements in a bin in order to maximize a maximal rectangle over the void space
- ✓ As a measure of extensibility, the following formula is proposed

$$E(s) = 1 - U(s) - \frac{w_{max}}{W} \cdot \frac{h_{max}}{H}$$

REORDERING FOR EXTENSIBILITY

- ✓ The goal is to re-order the elements in a bin in order to maximize a maximal rectangle over the void space
- ✓ As a measure of extensibility, the following formula is proposed

$$E(s) = 1 - U(s) - \frac{w_{max}}{W} \cdot \frac{h_{max}}{H}$$

- ✓ Reordering the elements in a bin with the proposed metric is not possible with a complete method like an ILP since the objective function is not linear

REORDERING FOR EXTENSIBILITY

- ✓ The goal is to re-order the elements in a bin in order to maximize a maximal rectangle over the void space
- ✓ As a measure of extensibility, the following formula is proposed

$$E(s) = 1 - U(s) - \frac{w_{max}}{W} \cdot \frac{h_{max}}{H}$$

- ✓ Reordering the elements in a bin with the proposed metric is not possible with a complete method like an ILP since the objective function is not linear
- ✓ An approach based on Simulated Annealing is used

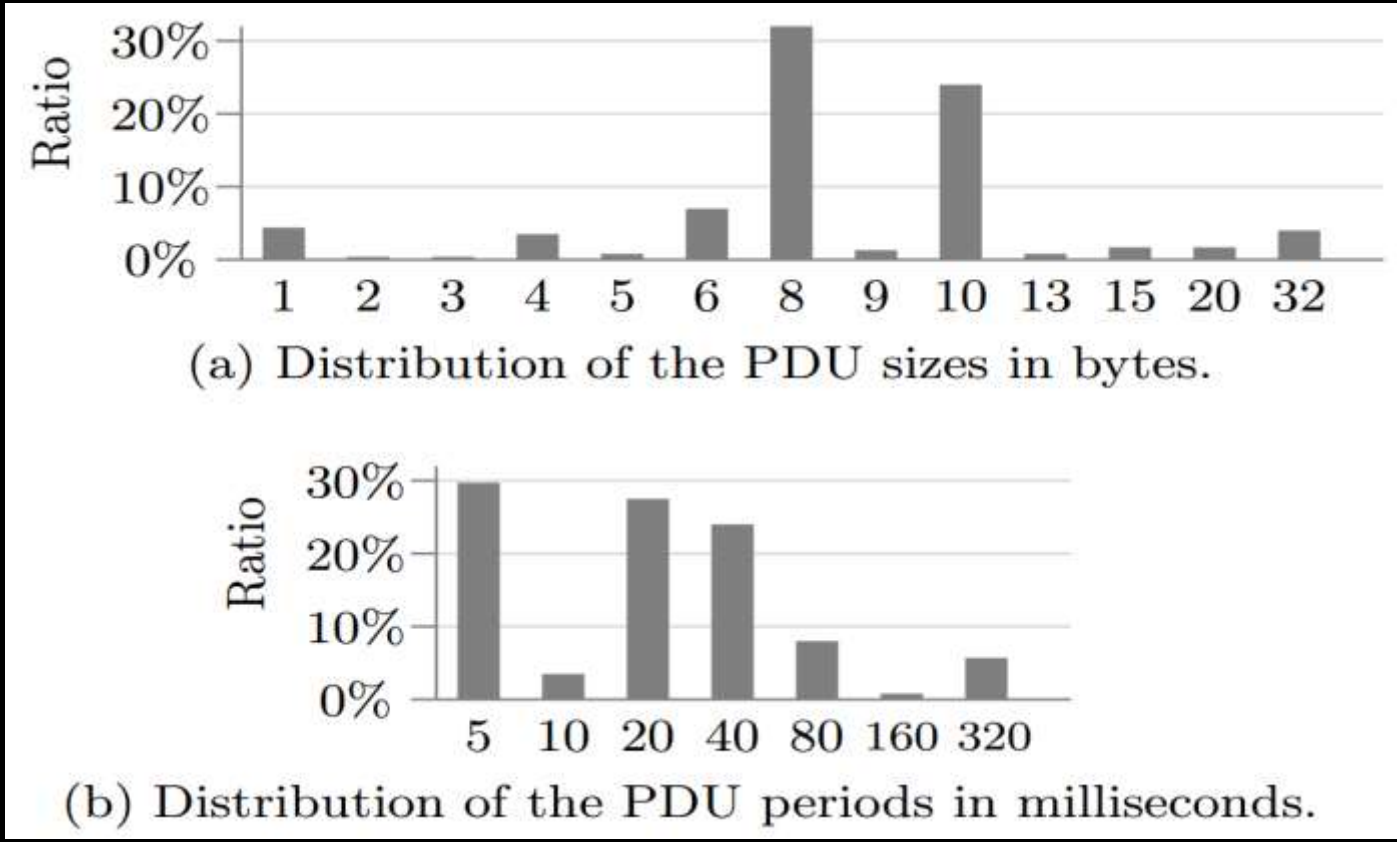
RESULTS

RESULTS

- ✓ PDU size and period distribution

RESULTS

- ✓ PDU size and period distribution



SCHEDULE OPTIMIZATION

SCHEDULE OPTIMIZATION

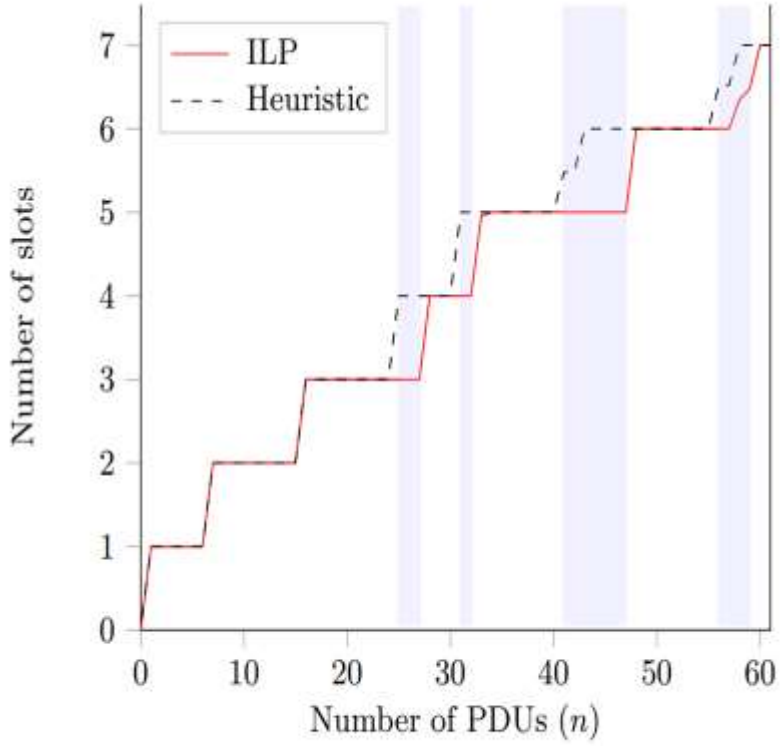
Method	runtime	slots	avg. $E(s)$
Heuristic	**0.065s**	**27**	0.029
+ Reordering	+42.5s	**27**	0.016
ILP* (CPLEX)	25.7s	**27**	0.028
ILP (CPLEX)	0.080s	**27**	0.028
+ Reordering	+48.0s	**27**	**0.019**
TTX Plan	360s	29	0.043

INCREMENTAL SCHEDULING

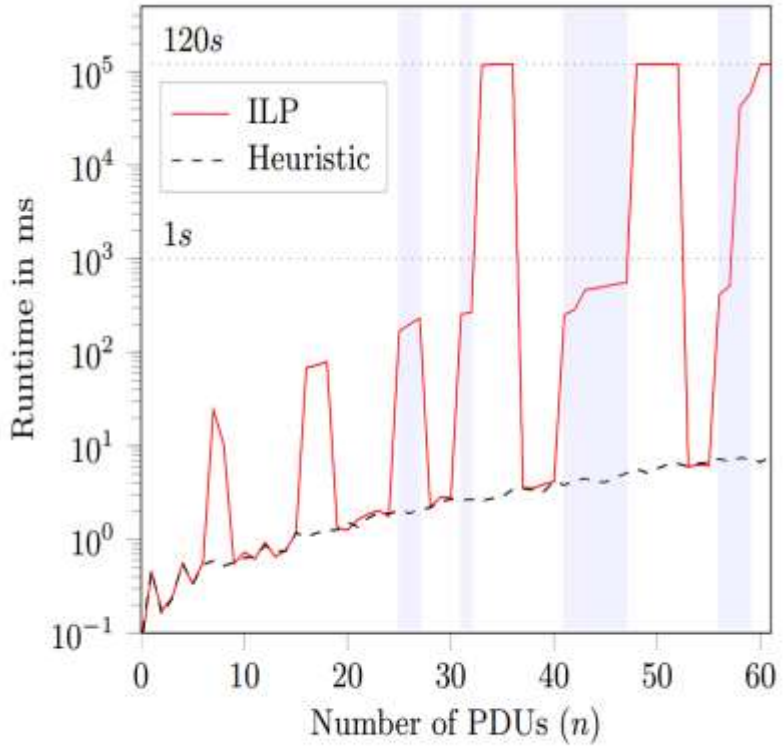
INCREMENTAL SCHEDULING

Method	runtime	slots	avg. $E(s)$
ILP (0.028) $\rightarrow$ Heuristic	0.022s	38	0.036
ILP (0.019) $\rightarrow$ Heuristic	0.020s	**37**	0.035
ILP (CPLEX)	12.2s	35	0.033

SCALABILITY



(a) Number of allocated slots depending on the number of scheduled PDUs.



(b) Runtime of the scheduling approaches depending on the number of scheduled PDUs.

THE END

THANK YOU