Real-Time Scheduling for 802.1Qbv Time-Sensitive Networking (TSN): A Systematic Review and Experimental Study

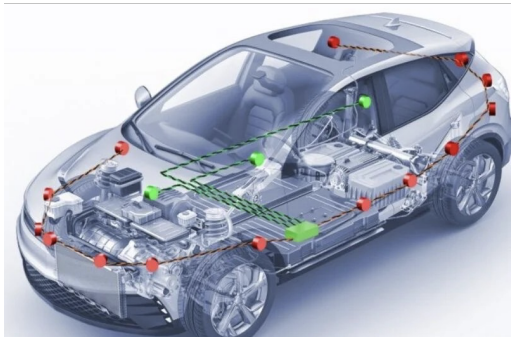
Chuanyu Xue[1], Tianyu Zhang[1], Yuanbin Zhou[2], Mark Nixon[3], Andrew Loveless[4], Song Han[1]

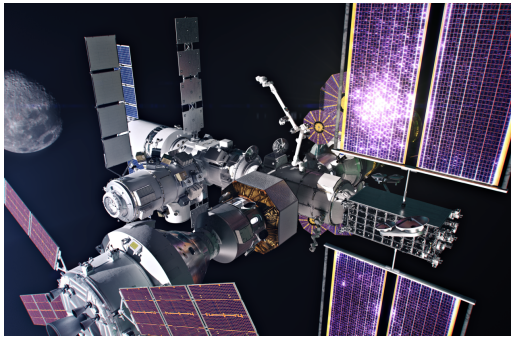
[1]School of Computing, University of Connecticut
[2]Singapore University of Technology and Design
[3]Emerson Automation Solutions
[4]NASA Johnson Space Center

Time-Sensitive Networking

Time-sensitive networking (TSN):

- IEEE 802.1 standards based on Ethernet.
- Provides deterministic Layer-2 communications.
- Essential for real-time industrial applications.

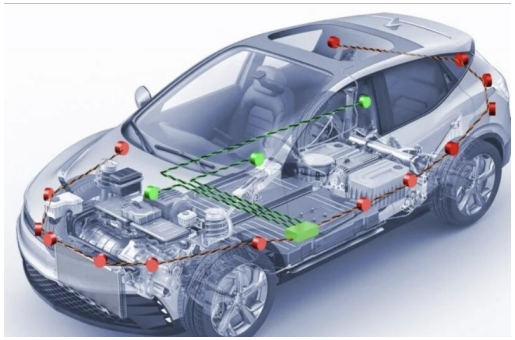


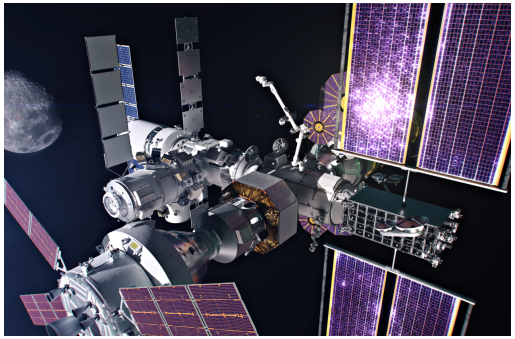


Time-Sensitive Networking

Time-sensitive networking (TSN):

- IEEE 802.1 standards based on Ethernet.
- Provides deterministic Layer-2 communications.
- Essential for real-time industrial applications.





Shapers and schedulers:

- Credit-Based Shaper (CBS), Strict-Priority Shaper (PS), Asynchronous Traffic Shaper (ATS), Time-Aware Shaper (TAS).
- Handle different traffic types and real-time requirements.

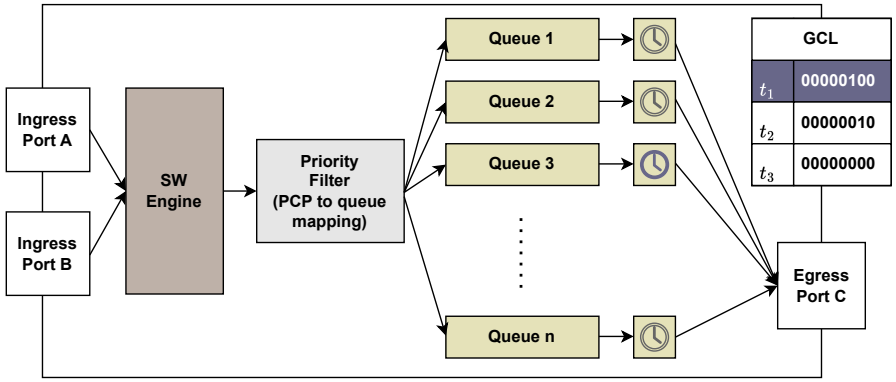
Time-Aware Shaper

Time-aware shaper (TAS) stands out in providing most stringent real-time performance guarantees, e.g., ultra-low latency and zero jitter.

Time-Aware Shaper

Time-aware shaper (TAS) stands out in providing most stringent real-time performance guarantees, e.g., ultra-low latency and zero jitter.

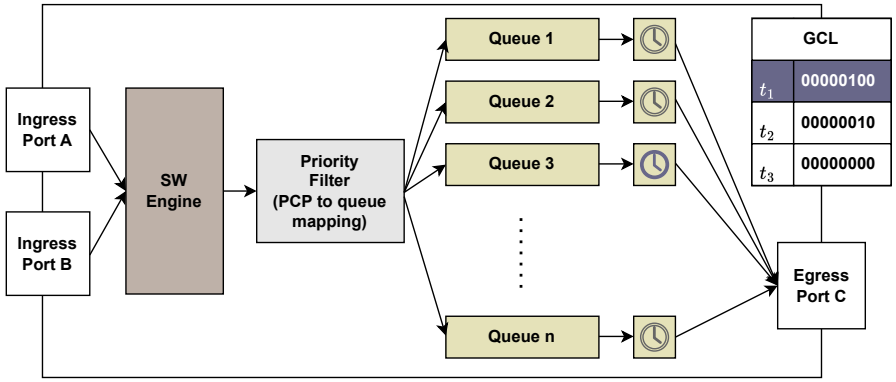
- Each bridge equipped with a set of time-gated queues and gate-control list (GCL).

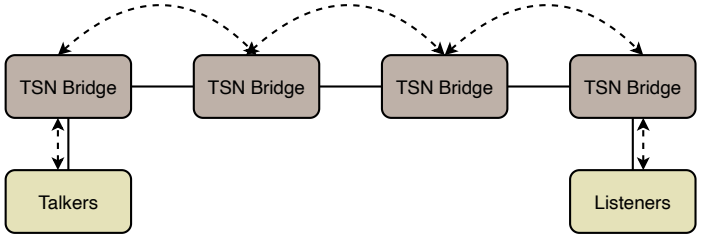


Time-Aware Shaper

Time-aware shaper (TAS) stands out in providing most stringent real-time performance guarantees, e.g., ultra-low latency and zero jitter.

- Each bridge equipped with a set of time-gated queues and gate-control list (GCL).
- Network-wide schedule generated by the Centralized Network Configuration (CNC).

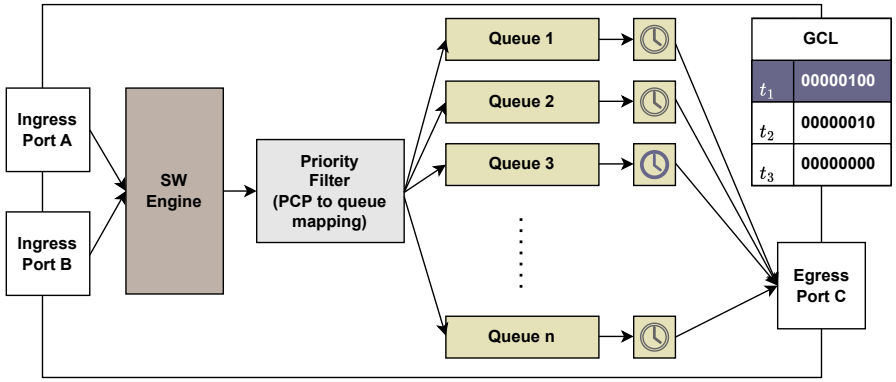


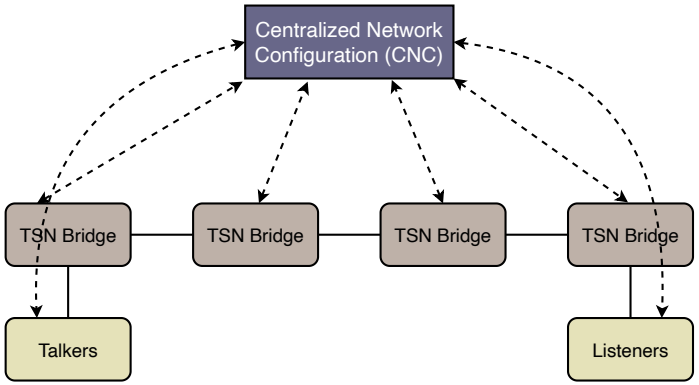


Time-Aware Shaper

Time-aware shaper (TAS) stands out in providing most stringent real-time performance guarantees, e.g., ultra-low latency and zero jitter.

- Each bridge equipped with a set of time-gated queues and gate-control list (GCL).
- Network-wide schedule generated by the Centralized Network Configuration (CNC).

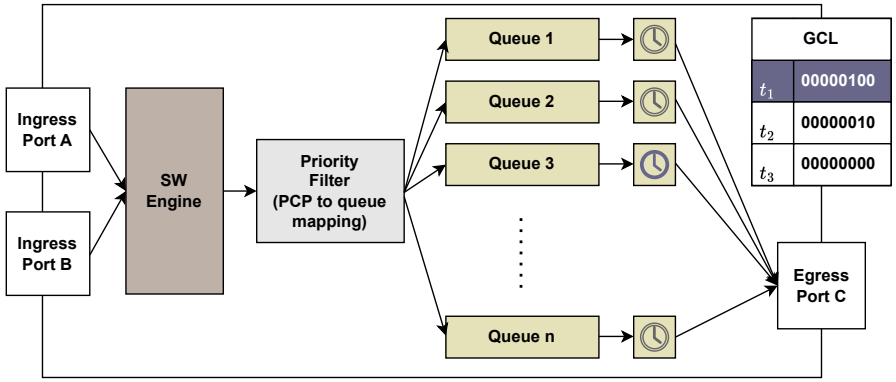


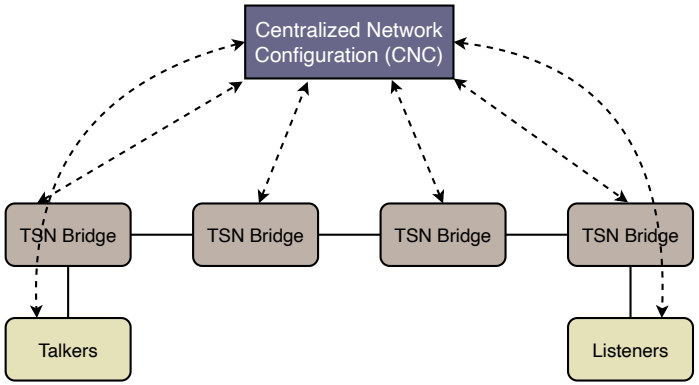


Time-Aware Shaper

Time-aware shaper (TAS) stands out in providing most stringent real-time performance guarantees, e.g., ultra-low latency and zero jitter.

- Each bridge equipped with a set of time-gated queues and gate-control list (GCL).
- Network-wide schedule generated by the Centralized Network Configuration (CNC).

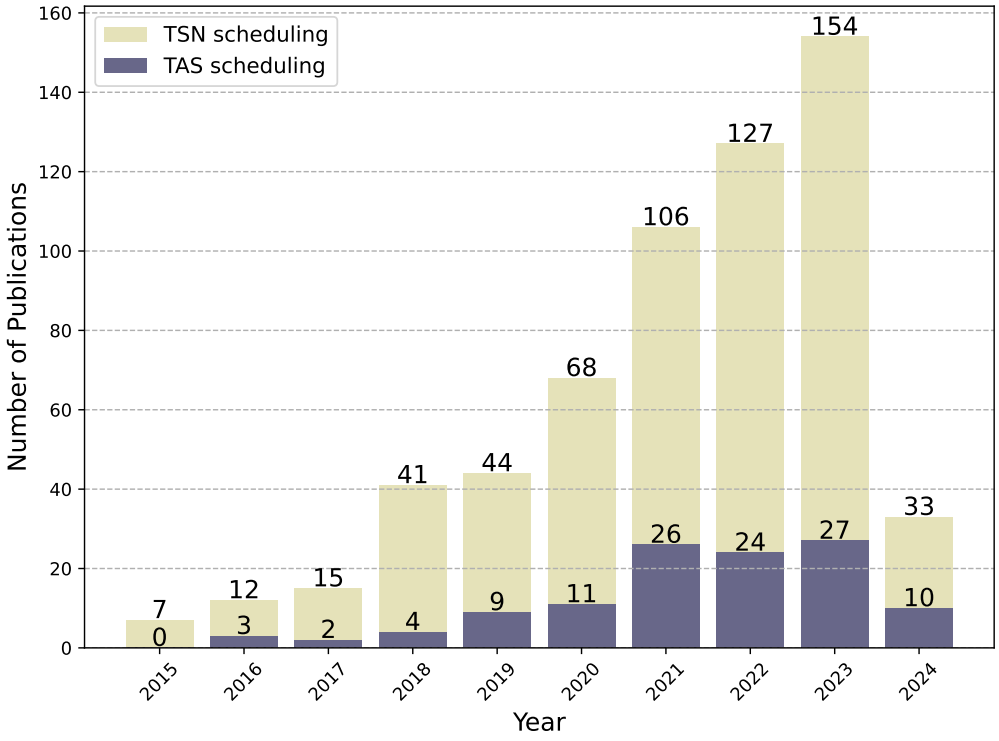




Generating a network-wide schedule for TAS-based network has no clear best practice

Existing Work

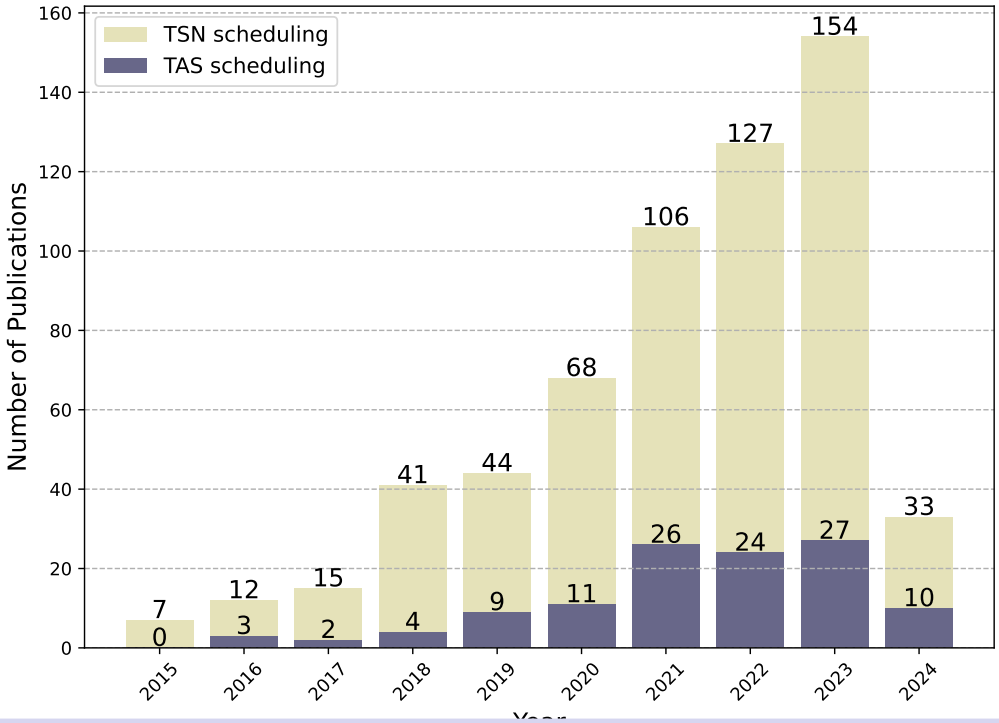
Number of publications about TAS scheduling from 2015 to 2024.



Search results for time-aware shaper from Sematic Scholar

Existing Work

Number of publications about TAS scheduling from 2015 to 2024.



Motivation

There is no comprehensive benchmark for TAS-based scheduling methods

Open Problems

Problem #1:
No systematic model-based categorization for TAS-based scheduling methods

Open Problems

Problem #1:
No systematic model-based categorization for TAS-based scheduling methods

Problem #2:
No comprehensive quantitative comparisons for TAS-based scheduling methods

Open Problems

Problem #1:
No systematic model-based categorization for TAS-based scheduling methods

Problem #2:
No comprehensive quantitative comparisons for TAS-based scheduling methods

Problem #3:
The correctness of existing methods has not been validated on real hardware

Contributions

1) Multi-dimensional model-based categorization

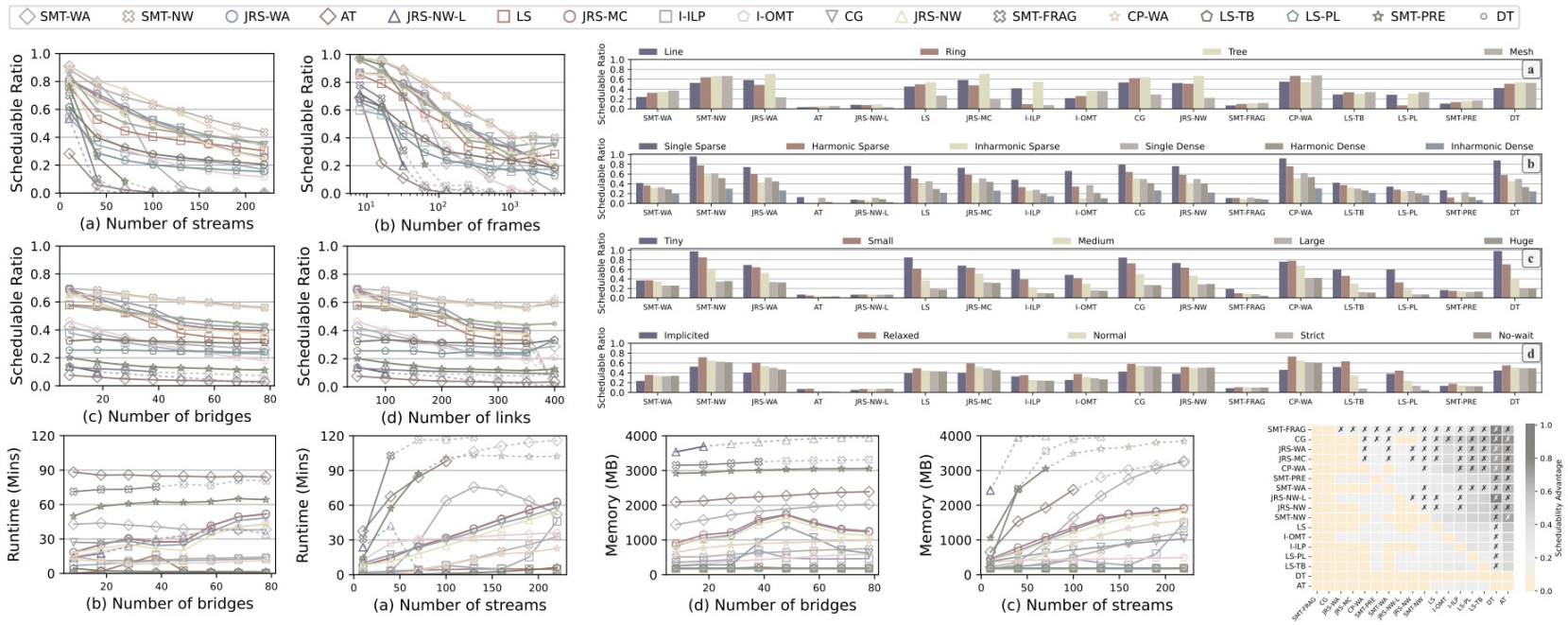
System models | Algorithms

17 methods

Article	Year	Fully schedulable	Strictly periodic	No-wait	Window -based	Queuing isolation	Routing	Multicast	Heuristic	Exact	Algorithms	Enhancements
Dürr et al. (SMT-NW)	2016	✓	✓	✓					(✓)	✓	ILP (Tabu)	
Craciunas et al. (SMT-WA)	2016	✓	✓			✓				✓	SMT	
Schweissguth et al. (JRS-WA)	2017	✓	✓				✓			✓	ILP	Paths reduce
Oliver et al. (AT)	2018	✓			✓	✓				✓	SMT	Array-theory
Falk et al. (JRS-NW-L)	2018	✓	✓	✓			✓			✓	ILP	Logic indicator
Pahlevan et al. (LS)	2019	✓	✓	✓			✓		✓		List scheduler	
Schweissguth et al. (JRS-MC)	2020	✓	✓				✓	✓		✓	ILP	Paths reduce
Atallah et al. (I-ILP)	2020	✓	✓	✓			✓	✓	✓		Iterative-ILP	
Jin et al. (I-OMT)	2020				✓	✓			✓	(✓)	Iterative-OMT	
Falk et al. (CG)	2020	✓	✓	✓			✓		✓	(✓)	Conflict-graph	
Hellmanns et al. (JRS-NW)	2021	✓	✓	✓			✓			✓	ILP	Path cut-off
Jin et al. (SMT-FRAG)	2021	✓	✓	✓					(✓)	✓	SMT (WCRT)	Fragmentation
Vlk et al. (CP-WA)	2021	✓	✓			✓			(✓)	✓	CP (Decompose)	
Vlk et al. (LS-TB)	2022		✓			✓			✓	(✓)	List scheduler	Traceback
Bujosa et al. (LS-PL)	2022		✓			✓			✓		List scheduler	Per-link search
Zhou et al. (SMT-PRE)	2022	✓								✓	SMT	Preemption
Zhang et al. (DT)	2022	✓	✓	✓					✓	(✓)	Divisibility	

Contributions

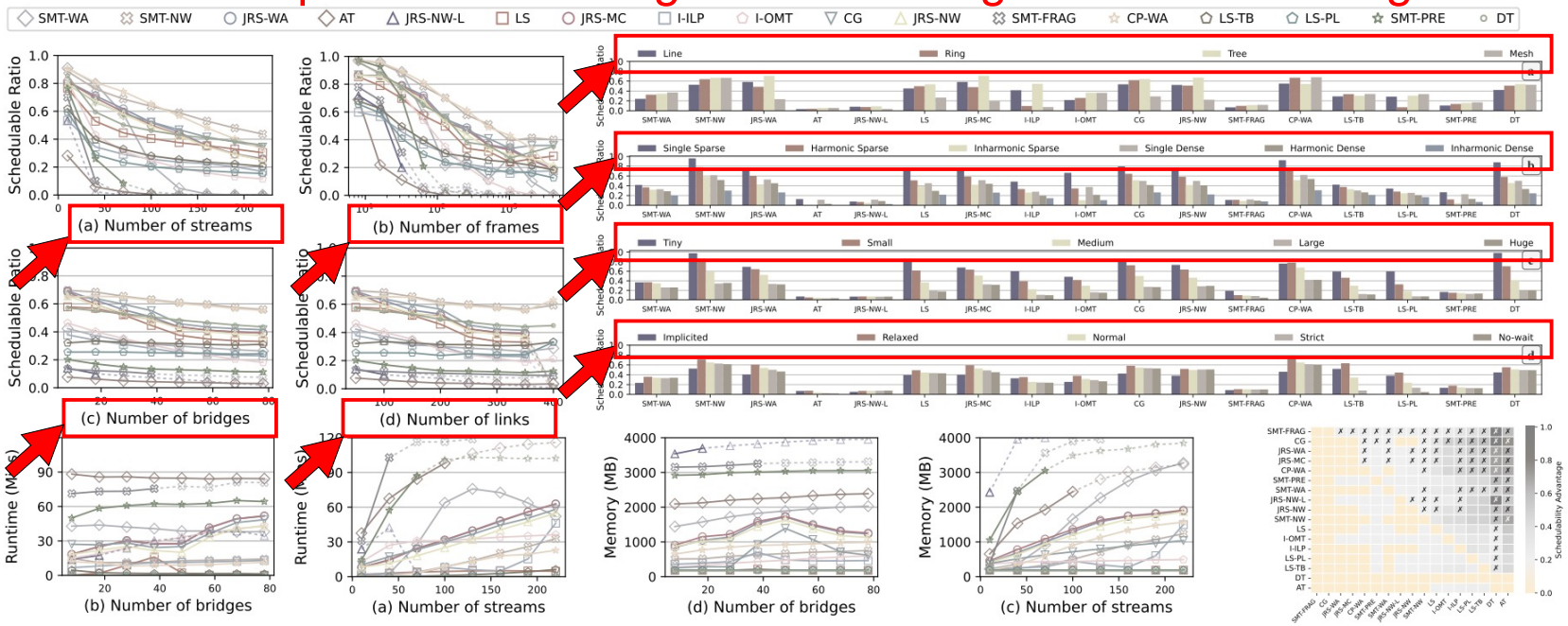
2) Benchmark in considering relevance, fairness, and reproducibility



Contributions

2) Benchmark in considering relevance, fairness, and reproducibility

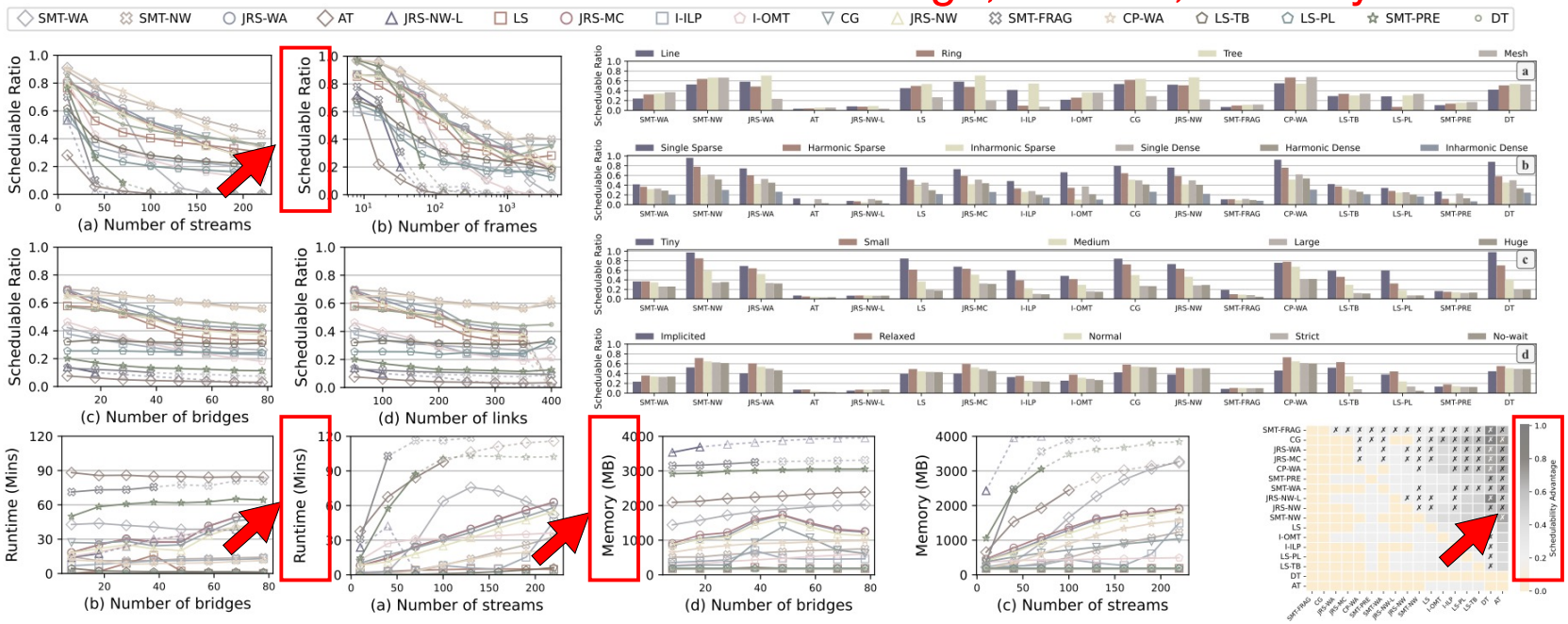
Extensive experimental settings: traffic settings + network settings



Contributions

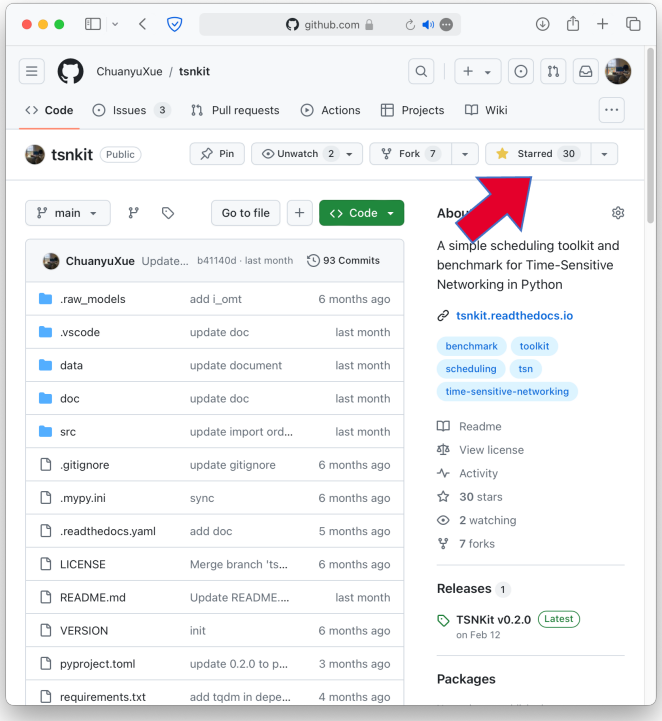
2) Benchmark in considering relevance, fairness, and reproducibility

Various metrics: schedulable ratio/advantage, runtime, memory



Contributions

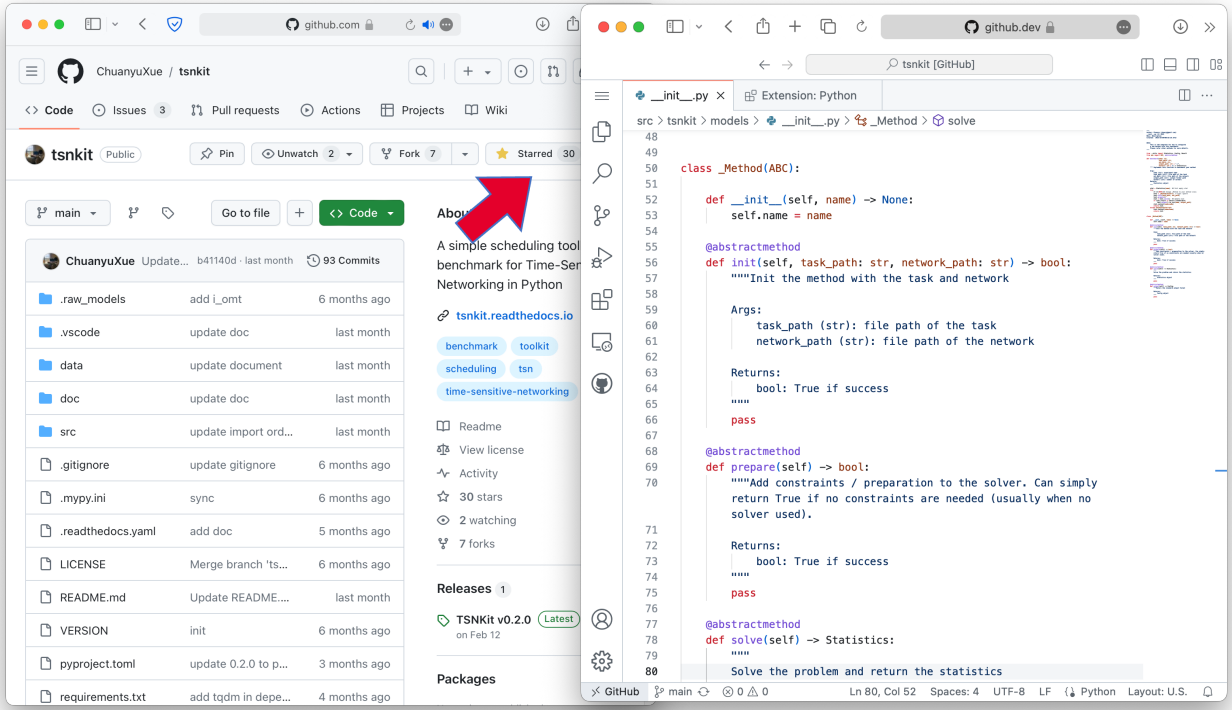
2) Benchmark in considering relevance, fairness, and reproducibility



Github page

Contributions

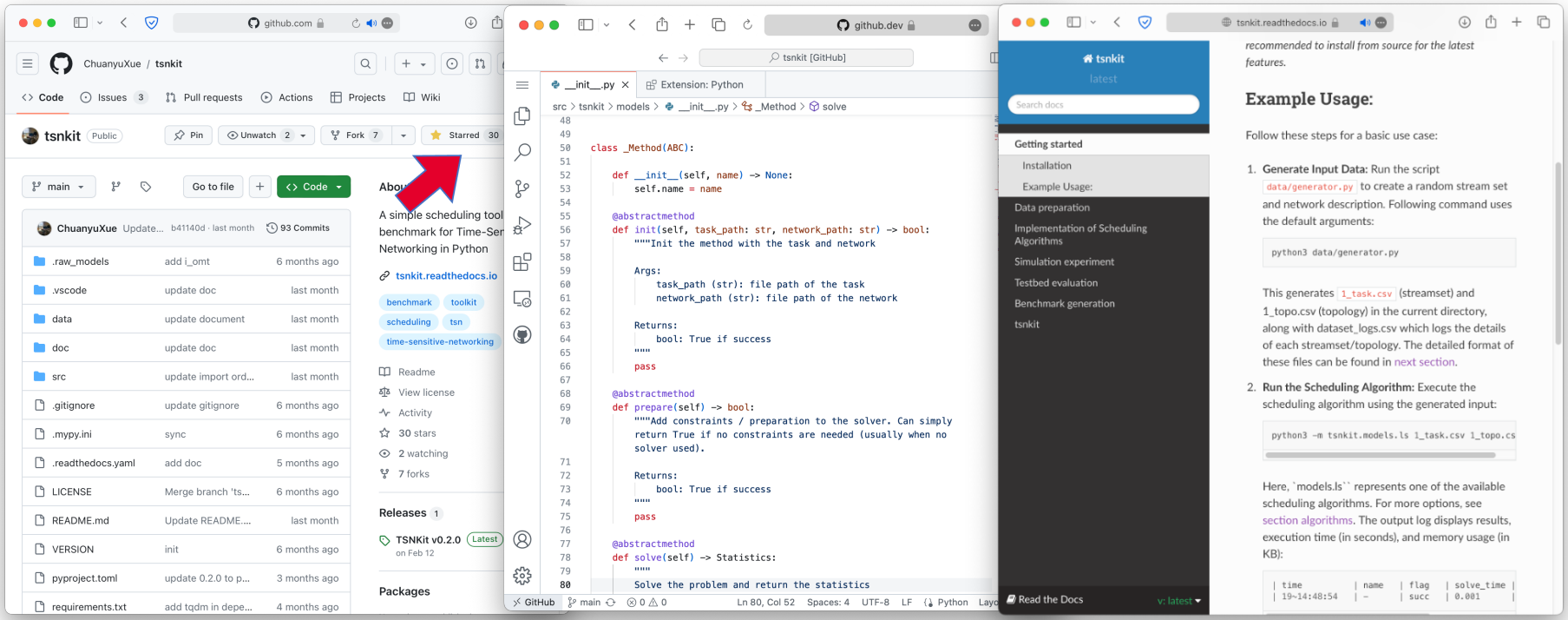
2) Benchmark in considering relevance, fairness, and reproducibility



Github page Extendable framework

Contributions

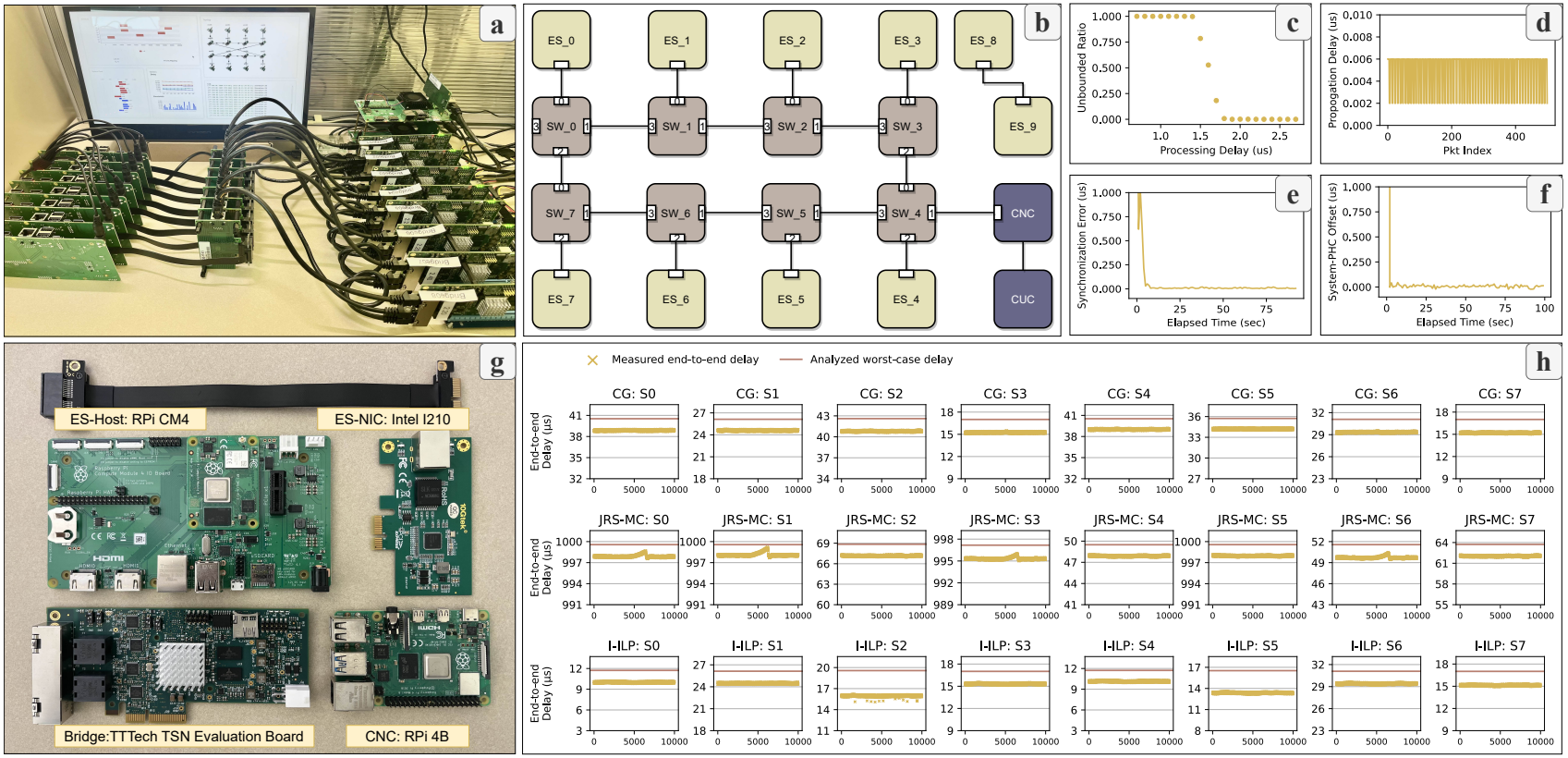
2) Benchmark in considering relevance, fairness, and reproducibility



Github page Extendable framework Documentation

Contributions

3) Real-world testbed for performance validation



Model-based Categorization

Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

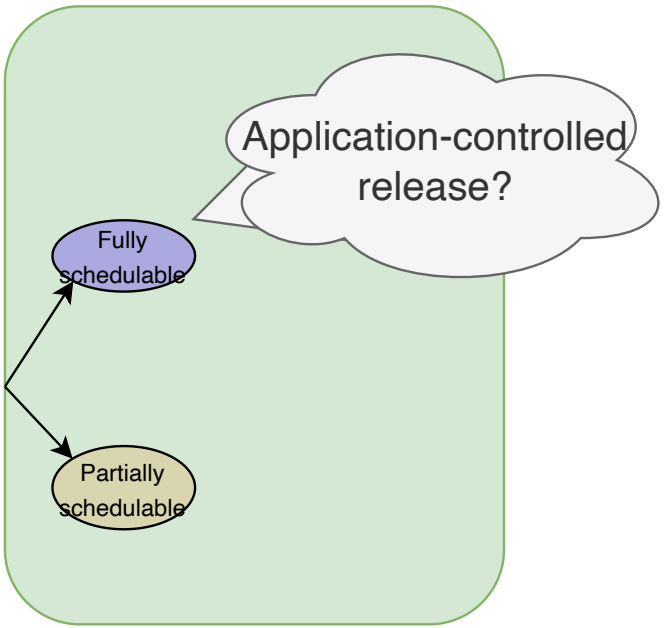
Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.

Model-based Categorization

Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.



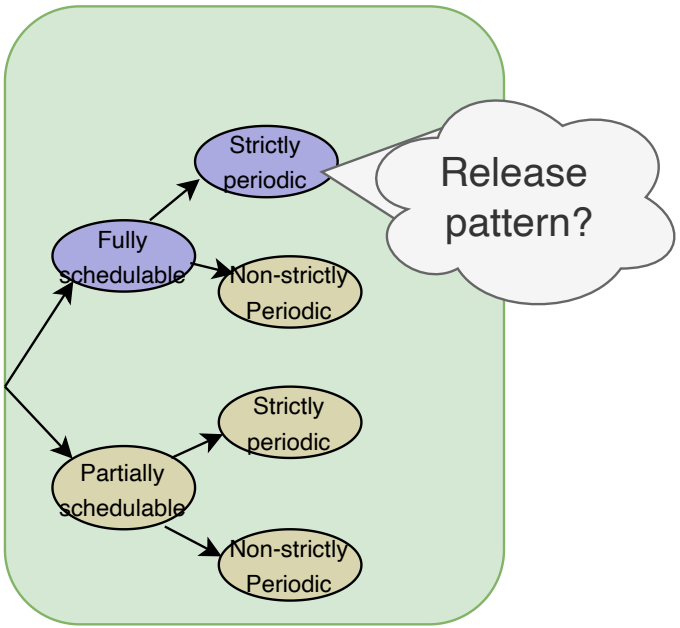
Traffic model

Model-based Categorization

Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.



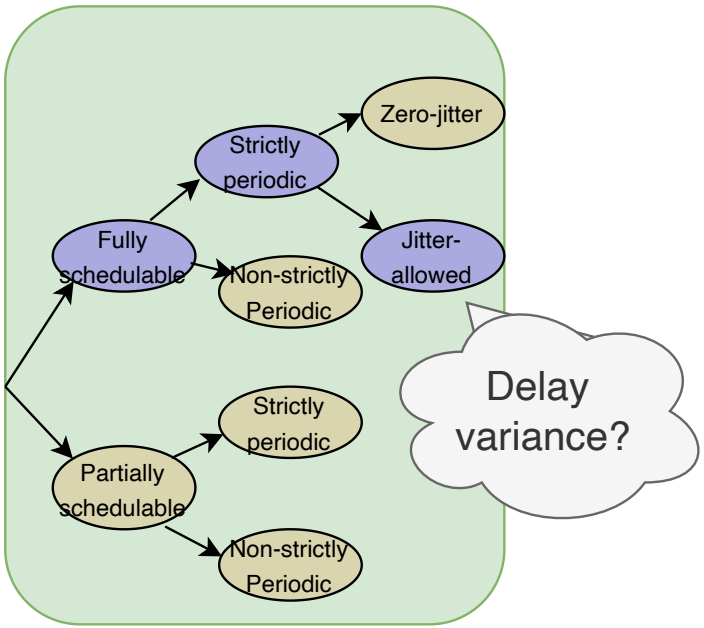
Traffic model

Model-based Categorization

Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.



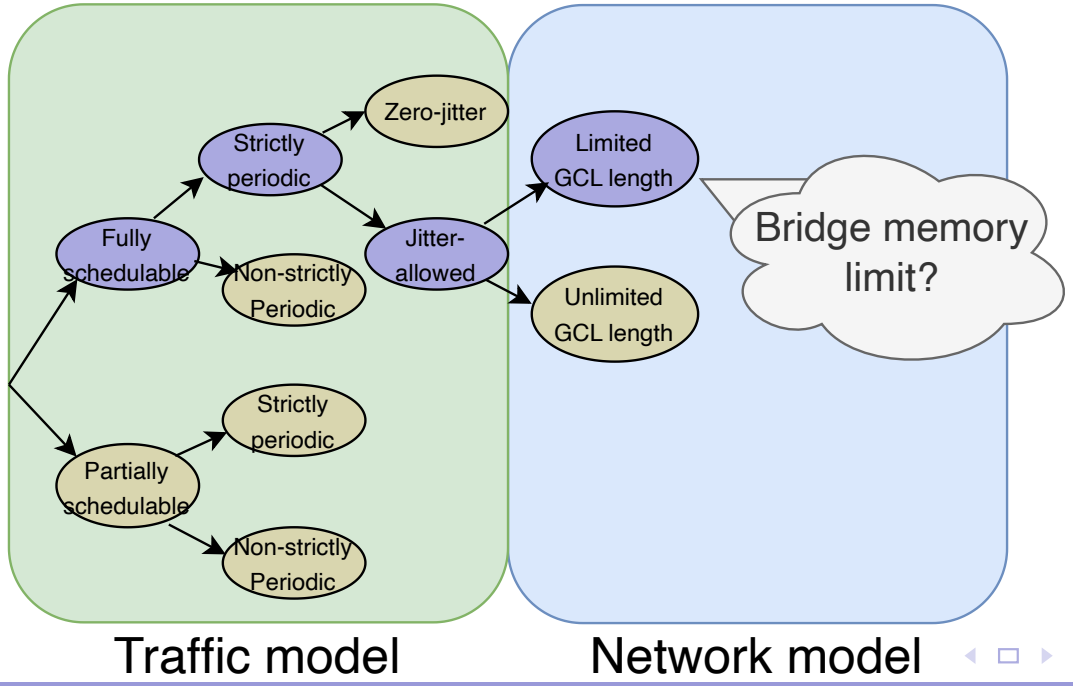
Traffic model

Model-based Categorization

Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.

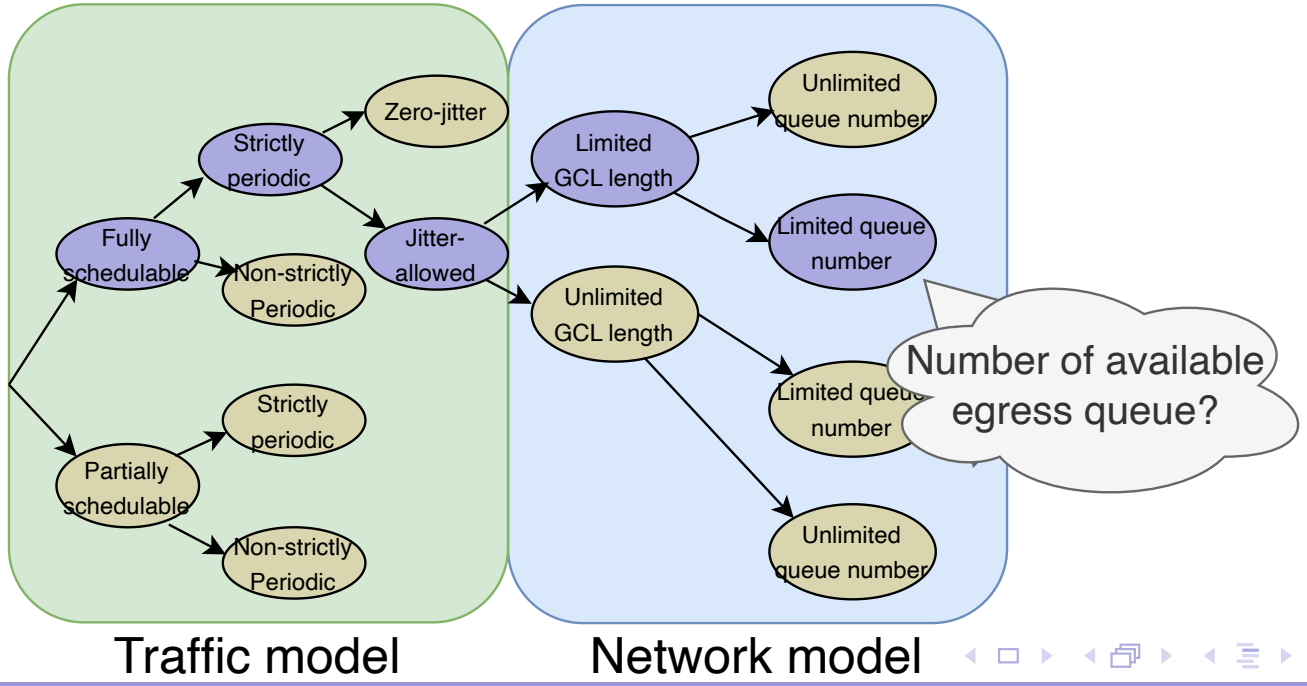


Model-based Categorization

Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.

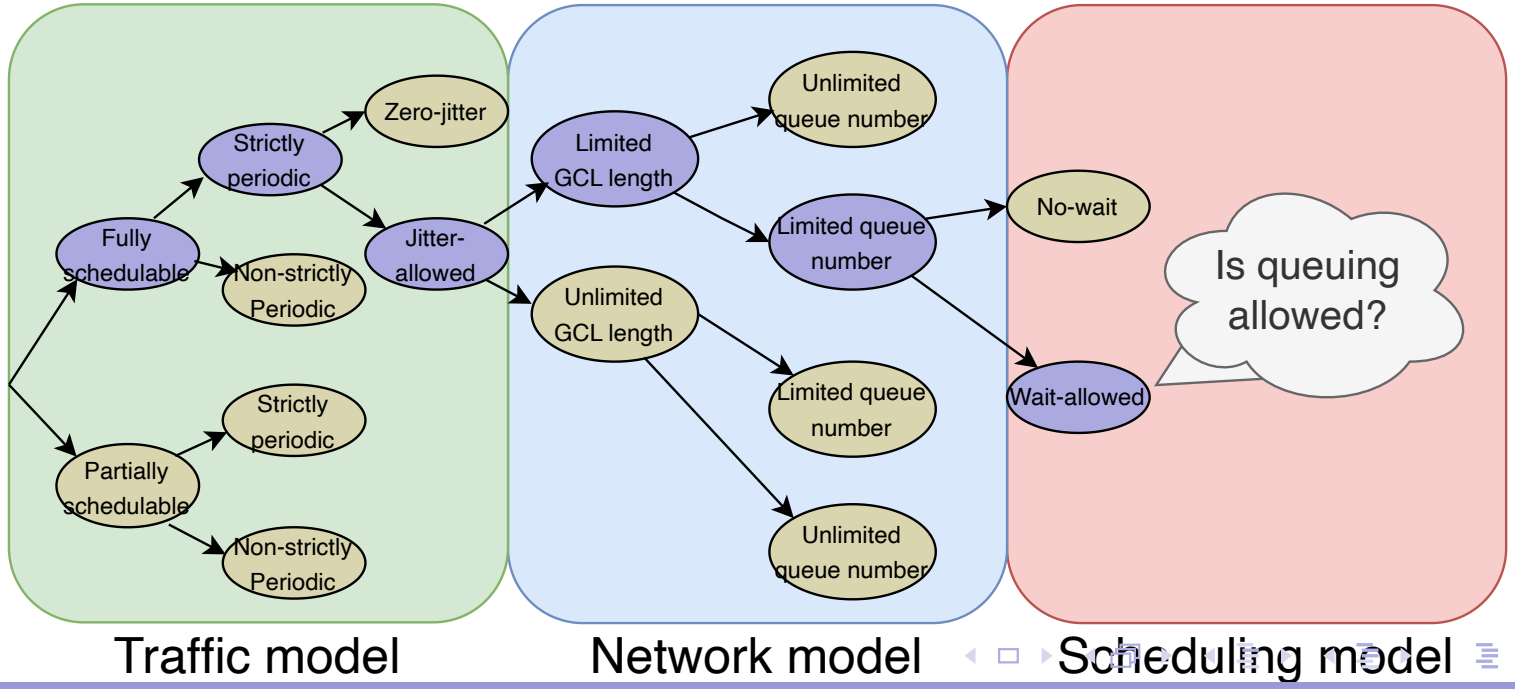


Model-based Categorization

Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.

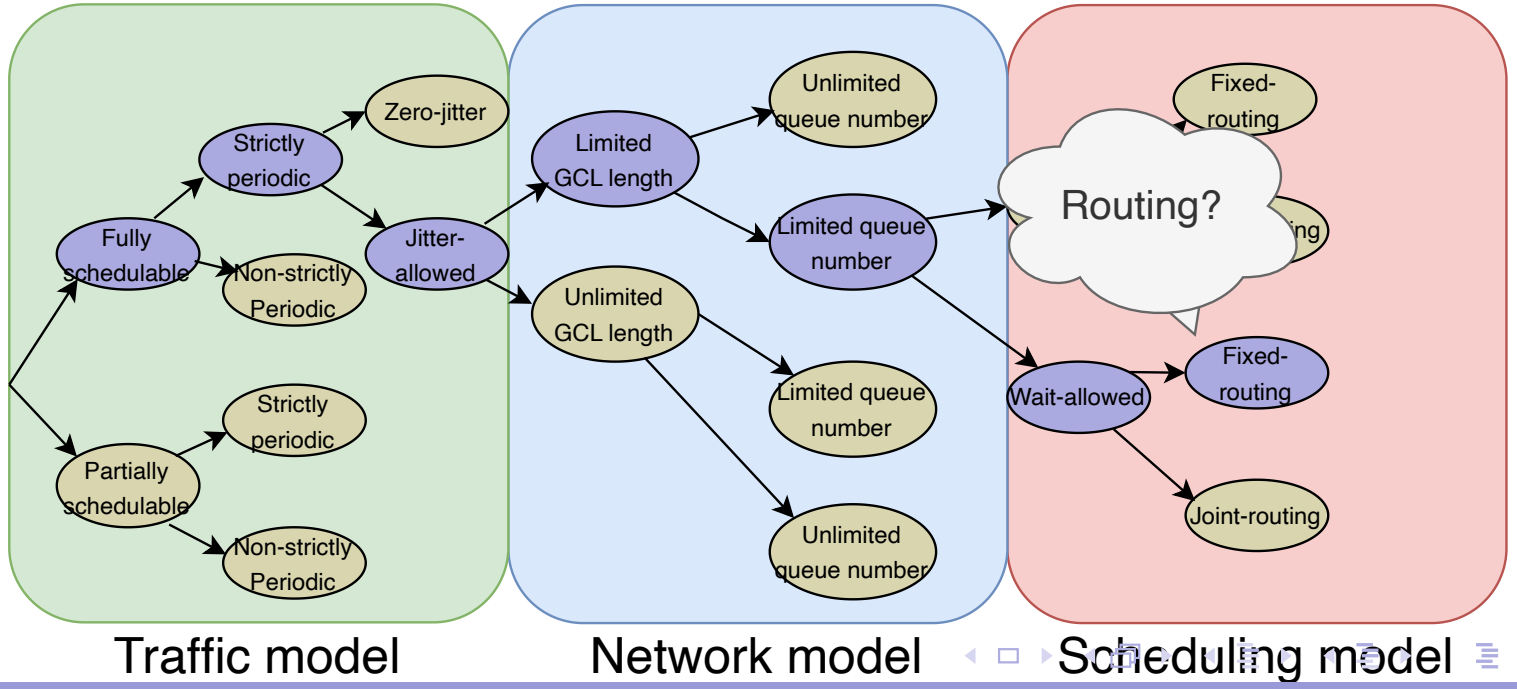


Model-based Categorization

Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.

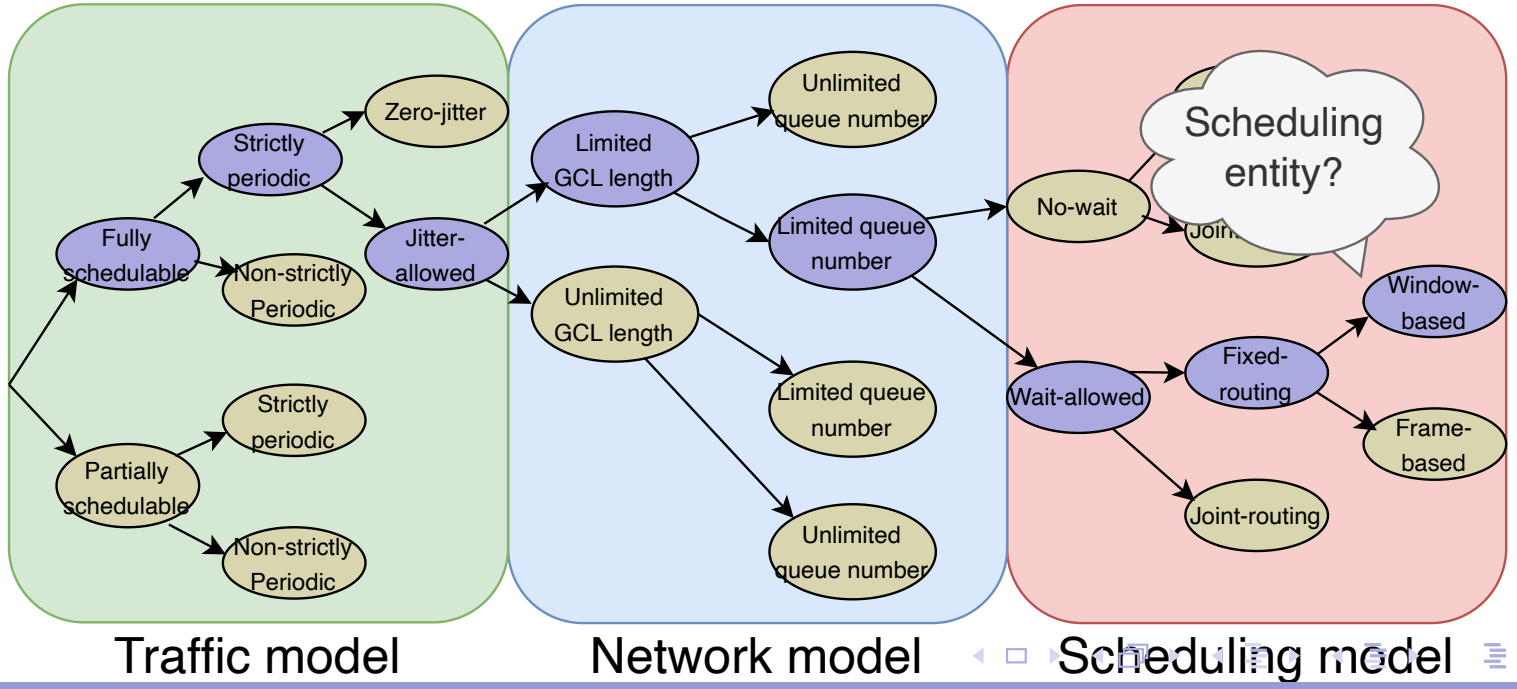


Model-based Categorization

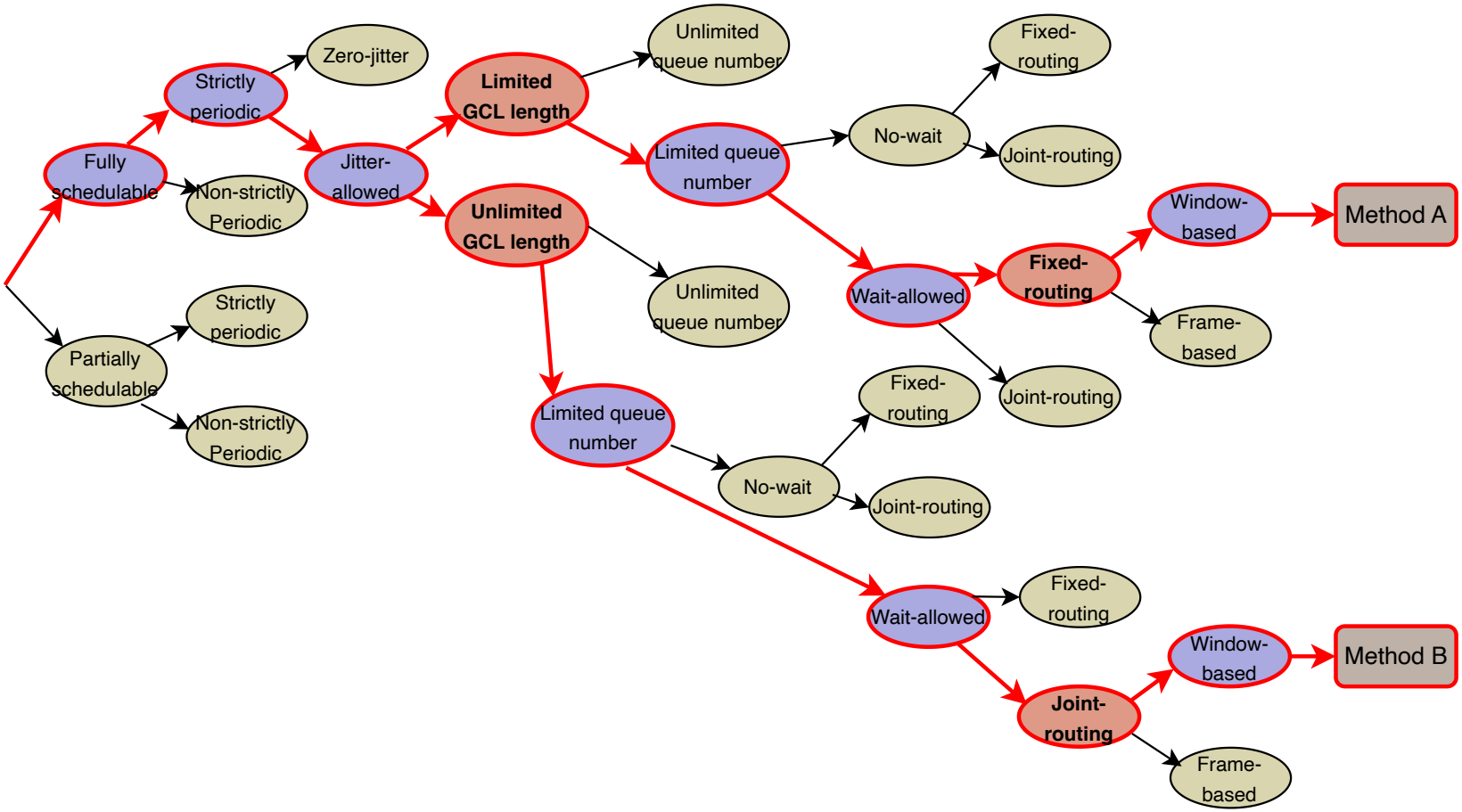
Motivation:

- For experimental study: Conduct model/algorithm-based experimental comparisons
- For the community: Quick identification of relevant methods

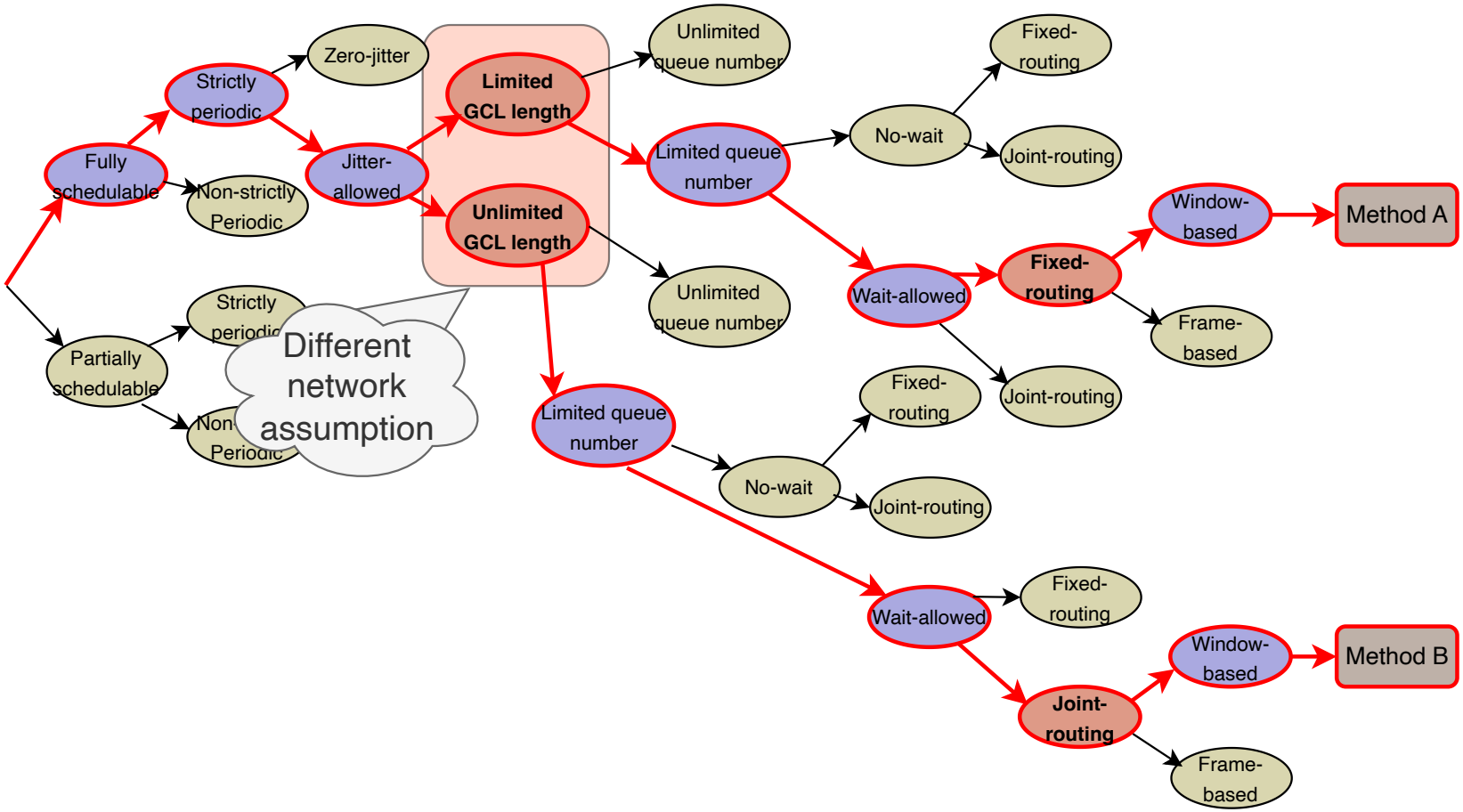
Principle: A 3-dimension model-based categorization to cover most existing TAS scheduling methods.



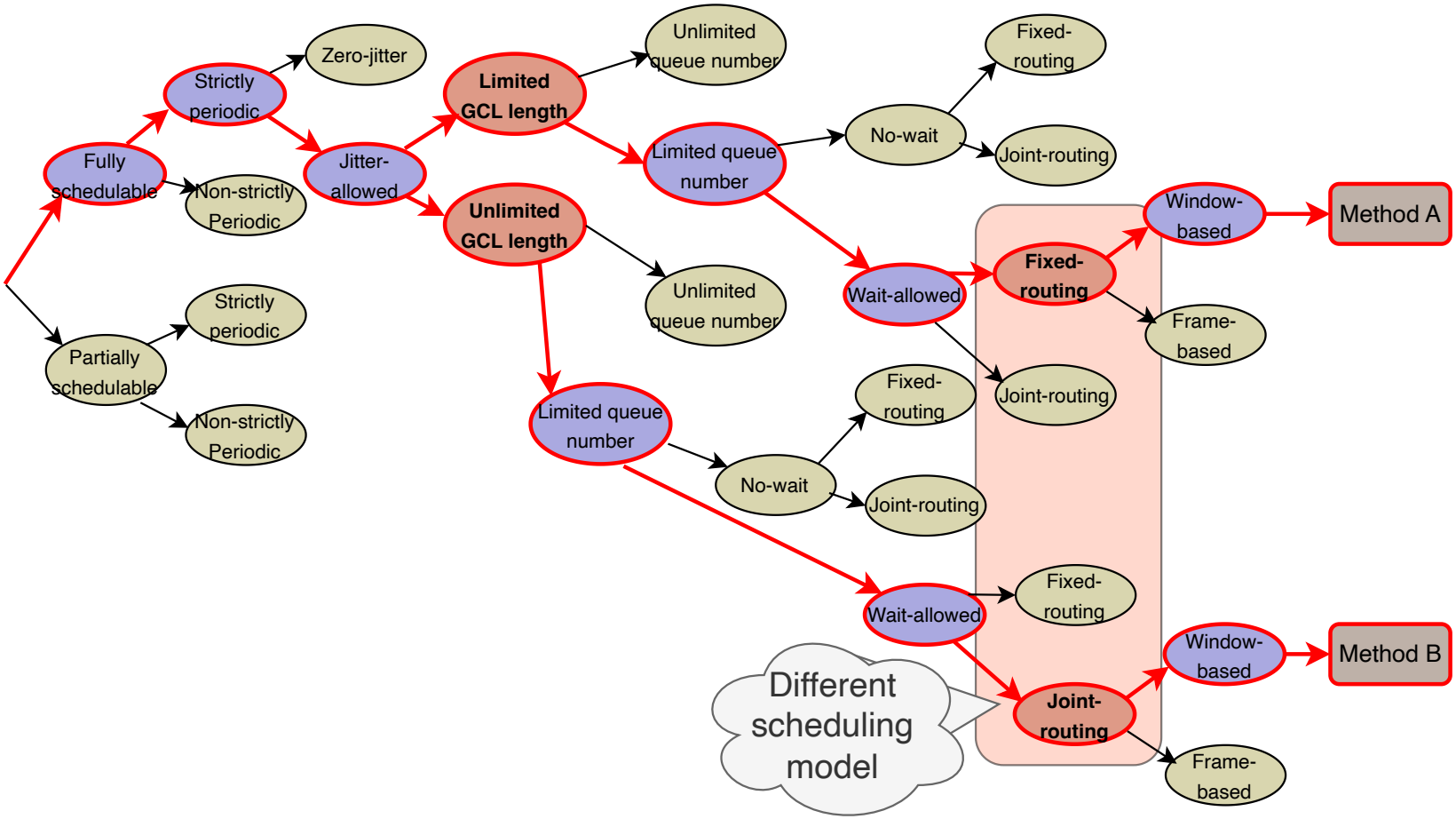
Model-based Categorization



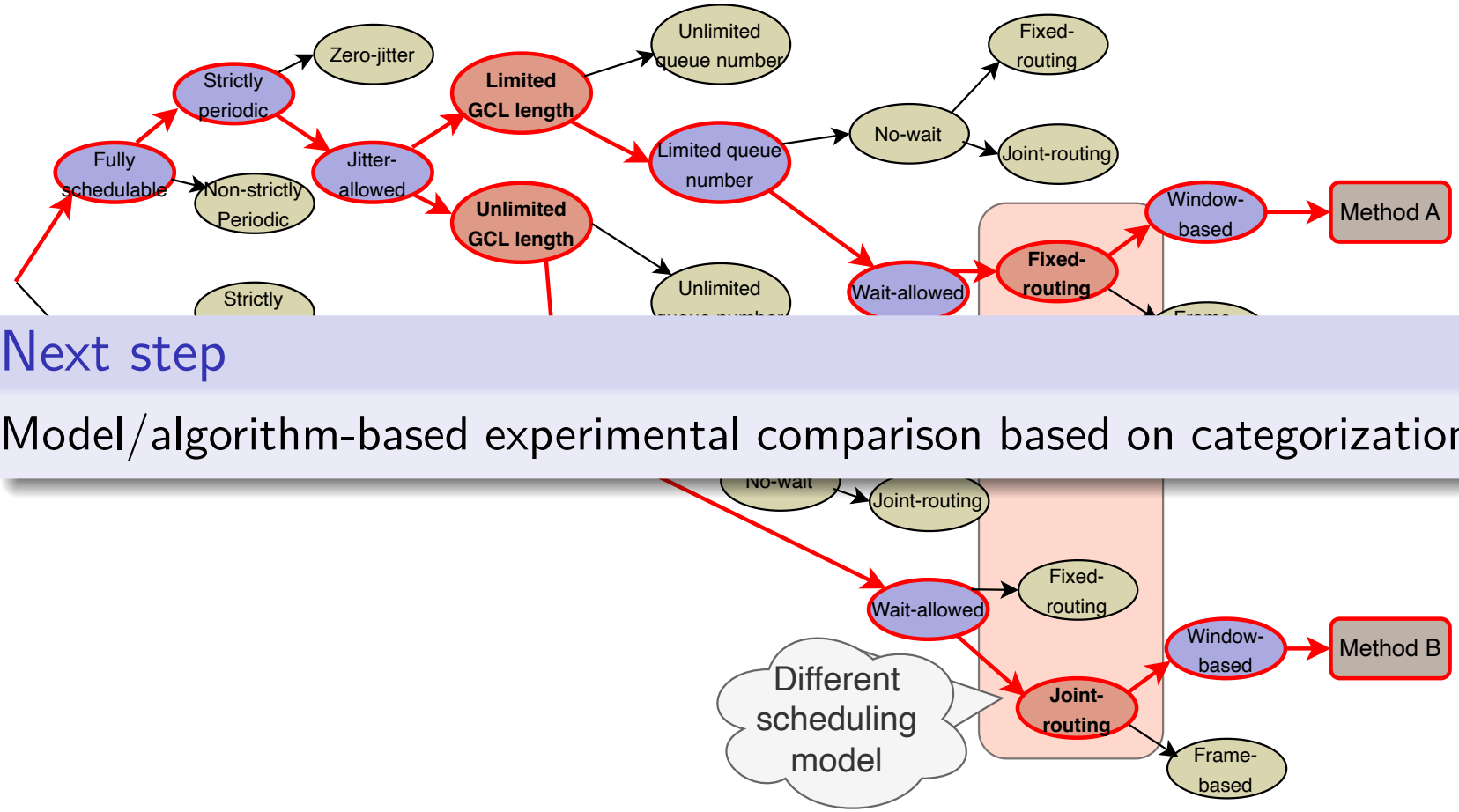
Model-based Categorization



Model-based Categorization



Model-based Categorization



Next step

Model/algorithm-based experimental comparison based on categorization.

Quantitative Comparisons

Motivation: Standardized benchmarks have become widely accepted tools in multiple fields of computer science and beyond, e.g., CPU/GPU, database, and machine/deep learning.

- Industry: Use the right tool for the right scenario
- Researchers: Understand the current limitations and potential research directions

Challenge

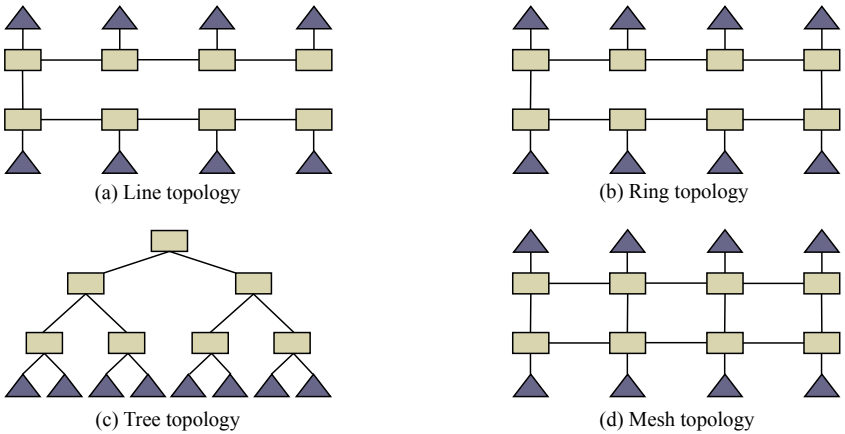
Design a high quality benchmark is a non-trivial task, it requires careful consideration of relevance, fairness, and reproducibility.

Relevance

Relevance: How closely the benchmark settings reflect the needs from the real-world application

- Include as board range of parameter settings as possible
- Reference to real-world scenario from standards or profiles, e.g., IEC/IEEE 60802, IEEE 802.1DP

	Parameter	Type	Value
Stream set	Number of streams	-	{10, 40, 70, . . . , 190, 220}
	Stream period (ms)	Sparse single	{2}
		Dense single	{0.4}
		Sparse harmonic	{0.5, 1, 2, 4}
		Dense harmonic	{0.1, 0.2, 0.4, 0.8}
		Sparse inharmonic	{0.25, 0.5, 1.25, 2.5, 4}
		Dense inharmonic	{0.05, 0.1, 0.25, 0.5, 0.8}
	Number of frames	-	{8, 16, 32, . . . , 2048, 4096}
	Stream payload (bytes)	Tiny size	50
		Small size	50 – 500
		Medium size	200 – 1500
		Large size	500 – 4500
		Extra size	1500 – 4500
	Stream deadline (ms)	Implicit deadline	Equal to period
		Relaxed deadline	NW + {0.1, 0.2, 0.4, 0.8, 1.6}
		Normal deadline	NW + {0.01, 0.025, 0.05, 0.1, 0.2, 0.4}
		Strict deadline	NW + {0, 0.01, 0.02, 0.025, 0.05}
		No-wait deadline	NW
Network	Topology	Linear, Ring, Tree, Mesh	
	Number of bridges	-	{8, 18, 28, . . . , 78}
	Number of links	-	{30, 32, 36, . . . , 386}
	Number of queues	-	{8}



Total 38,400 problem instances = traffic settings × network settings

Fairness

Timeout limit (2 hours): A common practice in experiment due to the NP-complete nature of TSN scheduling problem.

- Schedulable, unschedulable, unknown

Fairness

Timeout limit (2 hours): A common practice in experiment due to the NP-complete nature of TSN scheduling problem.

- Schedulable, unschedulable, unknown

Debias: Use two evaluation metrics for schedulability

- Metric 1: Scheduleable ratio (SR). Assume unknown results as unschedulable
  - Unfairness: Small timeout value $\rightarrow$ many unknown of methods require to run long time $\rightarrow$ low schedulability ratio

Fairness

Timeout limit (2 hours): A common practice in experiment due to the NP-complete nature of TSN scheduling problem.

- Schedulable, unschedulable, unknown

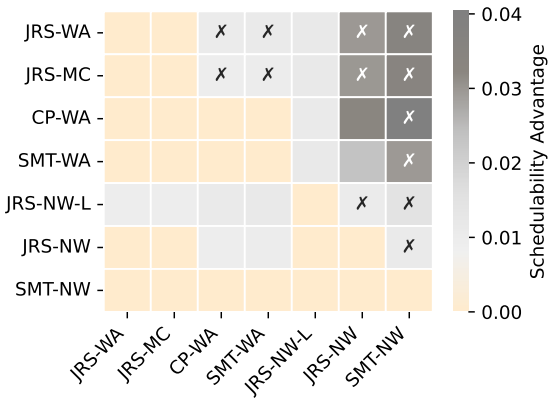
Debias: Use two evaluation metrics for schedulability

- Metric 1: Scheduleable ratio (SR). Assume unknown results as unschedulable
  - Unfairness: Small timeout value $\rightarrow$ many unknown of methods require to run long time $\rightarrow$ low schedulability ratio
- Metric 2: Schedulability advantage (SA). Novel pairwise performance metric without considering unknown
  - $\Phi(A, B)$: The number of instances that A is schedulable but B is unschedulable
  - $\Phi(A, B) > \Phi(B, A) = 0$: Method A dominates method B

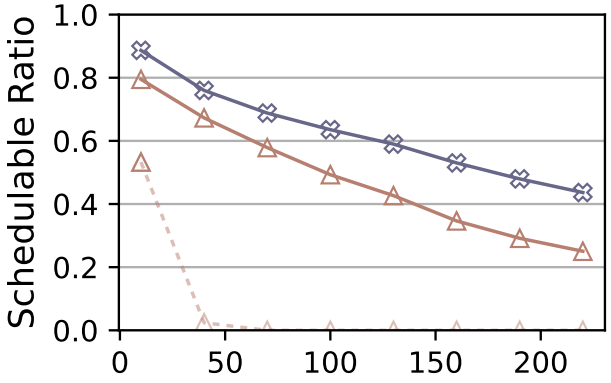
Model-based Experimental Study

Finding 1: Complex models may improve SA but be counterproductive on SR.

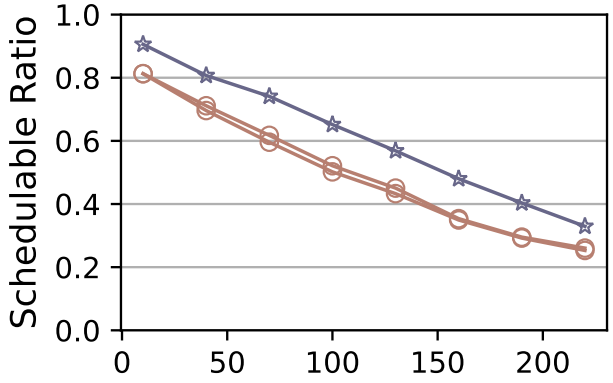
Example: Joint-routing vs. Fixed routing model (with both no-wait and wait-allowed models)



SA comparison



SR under no-wait model
FR-NW vs. JRS-NW, JRS-NW-L



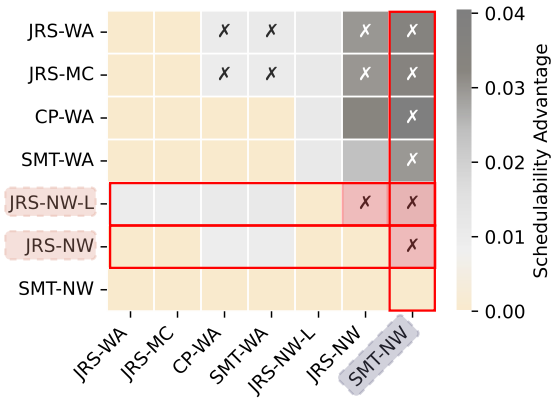
SR under wait-allowed model
CP-WA vs. JRS-NW, JRS-NW-L

Other inconsistent SR and SA: No-wait vs. wait-allowed, FRAG/PRE vs. non-FRAG/non-PRE

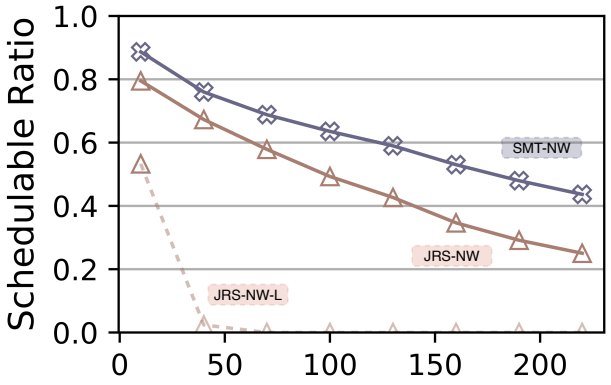
Model-based Experimental Study

Finding 1: Complex models may improve SA but be counterproductive on SR.

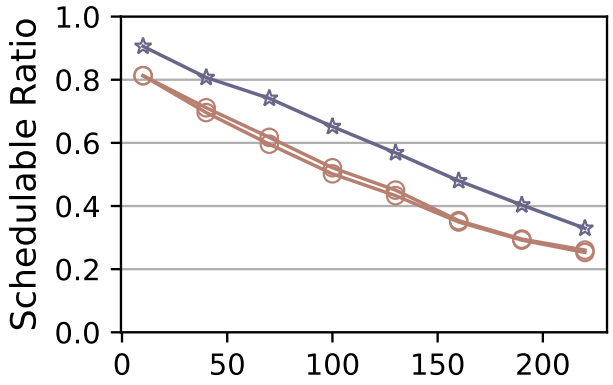
Example: Joint-routing vs. Fixed routing model (with both no-wait and wait-allowed models)



SA comparison



SR under no-wait model
FR-NW vs. JRS-NW, JRS-NW-L



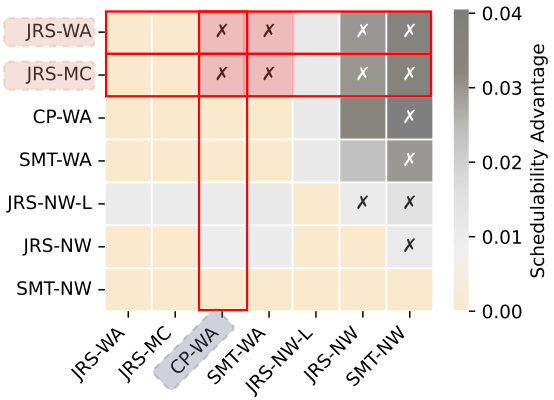
SR under wait-allowed model
CP-WA vs. JRS-NW, JRS-NW-L

Other inconsistent SR and SA: No-wait vs. wait-allowed, FRAG/PRE vs. non-FRAG/non-PRE

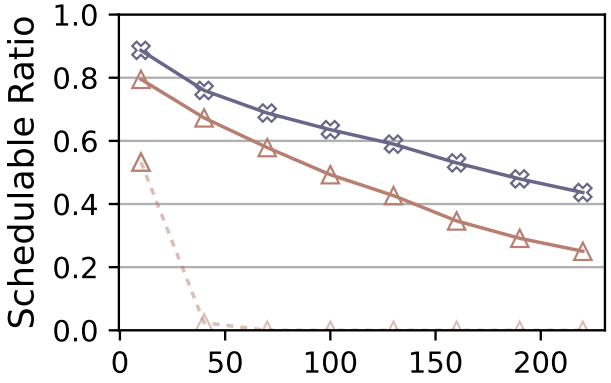
Model-based Experimental Study

Finding 1: Complex models may improve SA but be counterproductive on SR.

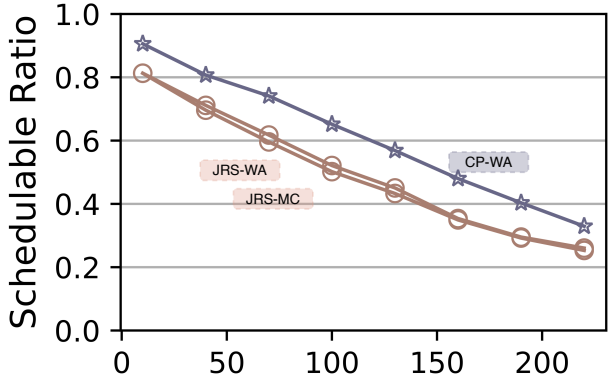
Example: Joint-routing vs. Fixed routing model (with both no-wait and wait-allowed models)



SA comparison



SR under no-wait model
FR-NW vs. JRS-NW, JRS-NW-L



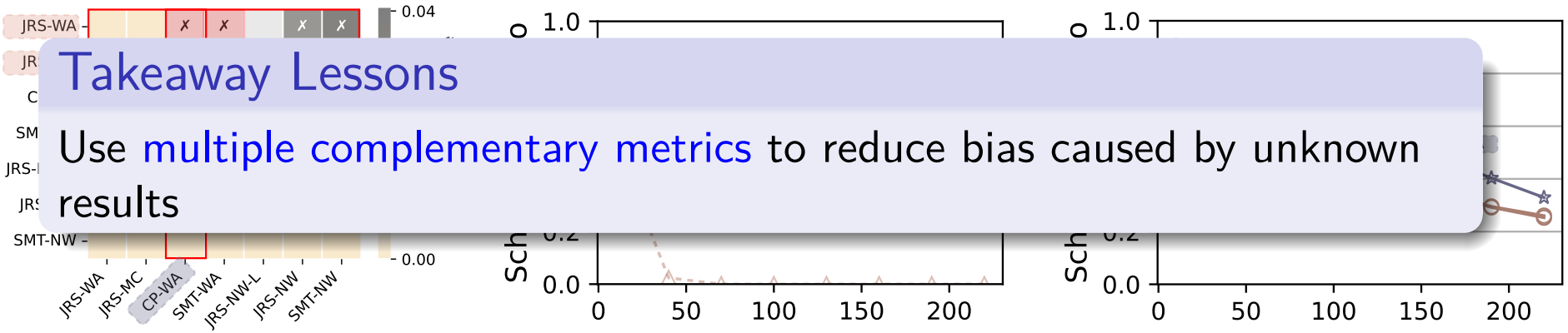
SR under wait-allowed model
CP-WA vs. JRS-NW, JRS-NW-L

Other inconsistent SR and SA: No-wait vs. wait-allowed, FRAG/PRE vs. non-FRAG/non-PRE

Model-based Experimental Study

Finding 1: Complex models may improve SA but be counterproductive on SR.

Example: Joint-routing vs. Fixed routing model (with both no-wait and wait-allowed models)



SA comparison

SR under no-wait model
FR-NW vs. JRS-NW, JRS-NW-L

SR under wait-allowed model
CP-WA vs. JRS-NW, JRS-NW-L

Takeaway Lessons

Use multiple complementary metrics to reduce bias caused by unknown results

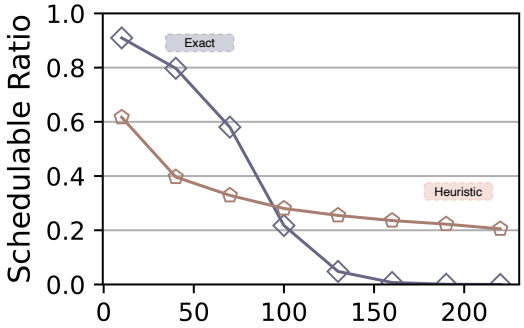
Other inconsistent SR and SA: No-wait vs. wait-allowed, FRAG/PRE vs. non-FRAG/non-PRE

Algorithm-based Experimental Study

Finding 2: Schedulability varies under different settings, even using the same evaluation metric.

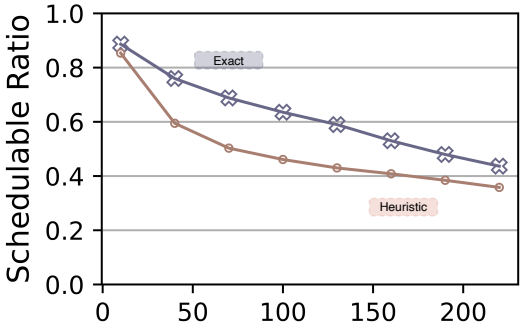
- Heuristic performs similar/better than exact algorithm in large-scale

FR and wait-allowed



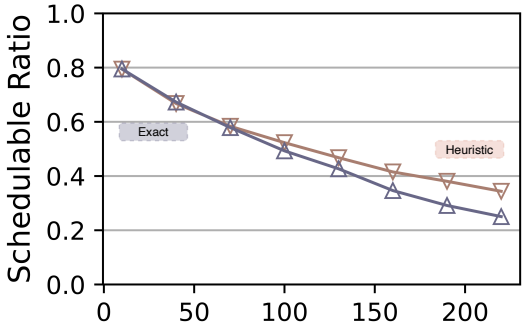
SMT-WA vs. LS-TB

FR and no-wait



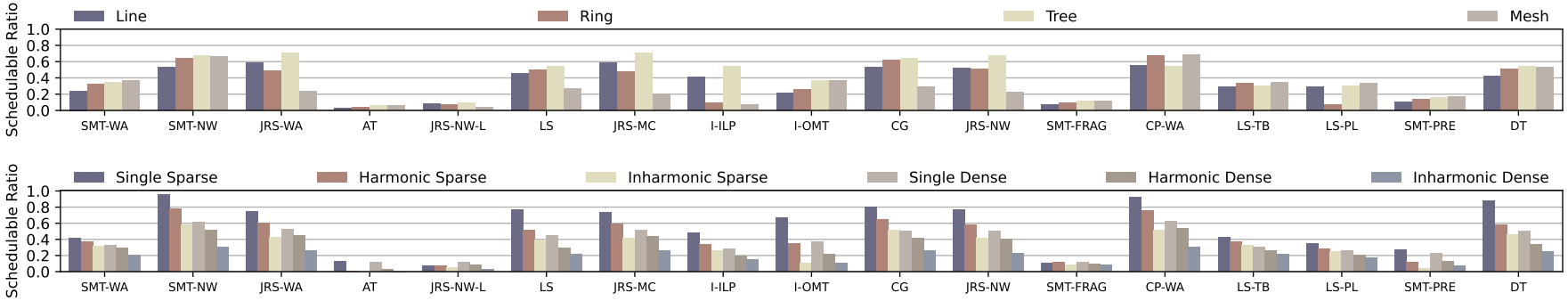
SMT-NW vs. DT

JRS and no-wait



JRS-NW vs. CG

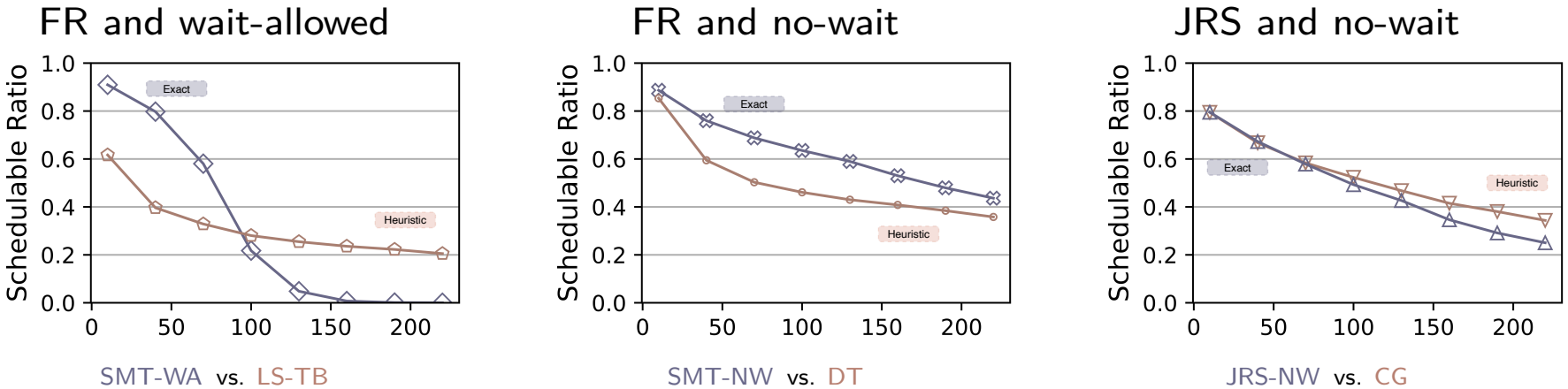
- Some heuristic methods suffer under certain settings



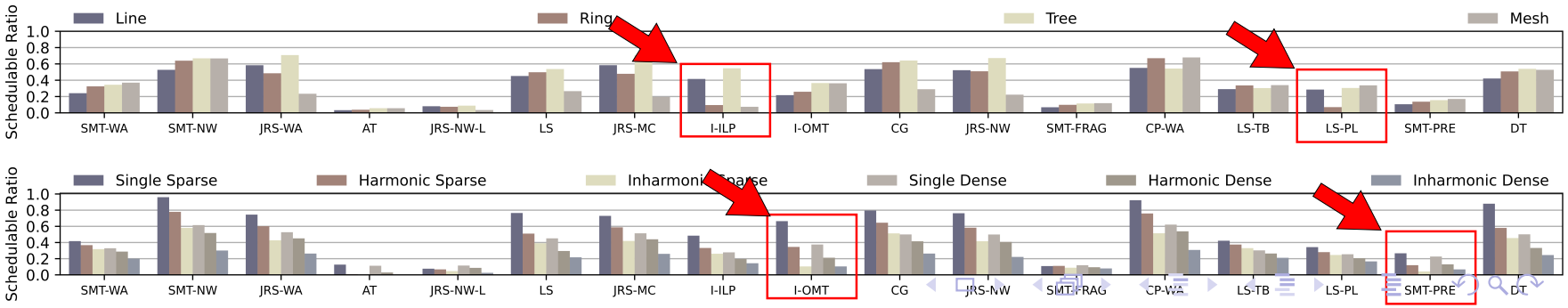
Algorithm-based Experimental Study

Finding 2: Schedulability varies under different settings, even using the same evaluation metric.

- Heuristic performs similar/better than exact algorithm in large-scale



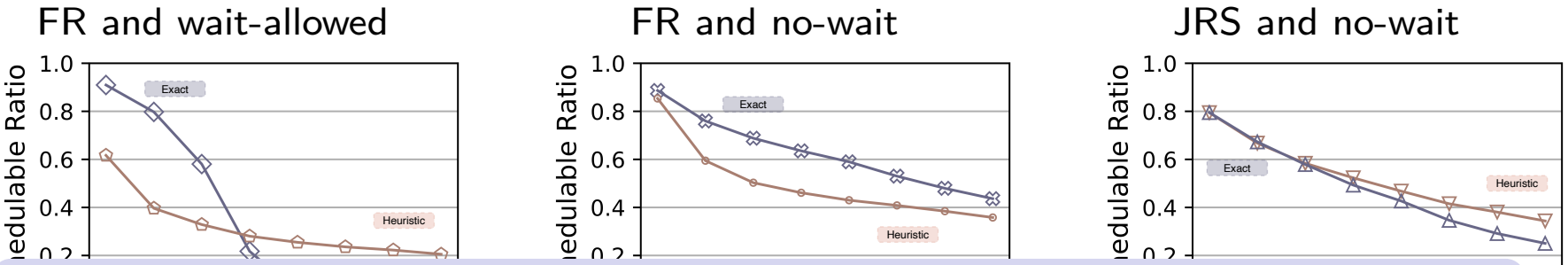
- Some heuristic methods suffer under certain settings



Algorithm-based Experimental Study

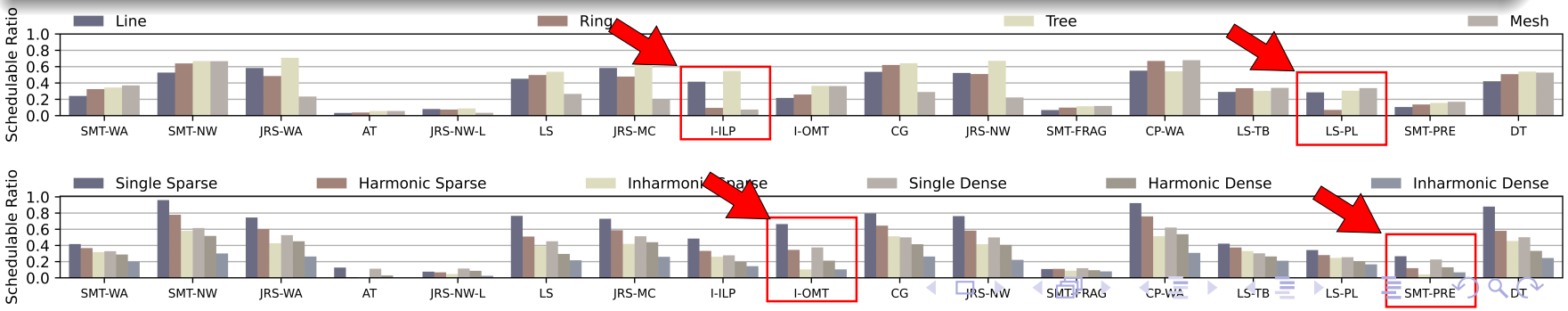
Finding 2: Schedulability varies under different settings, even using the same evaluation metric.

- Heuristic performs similar/better than exact algorithm in large-scale



Takeaway Lessons

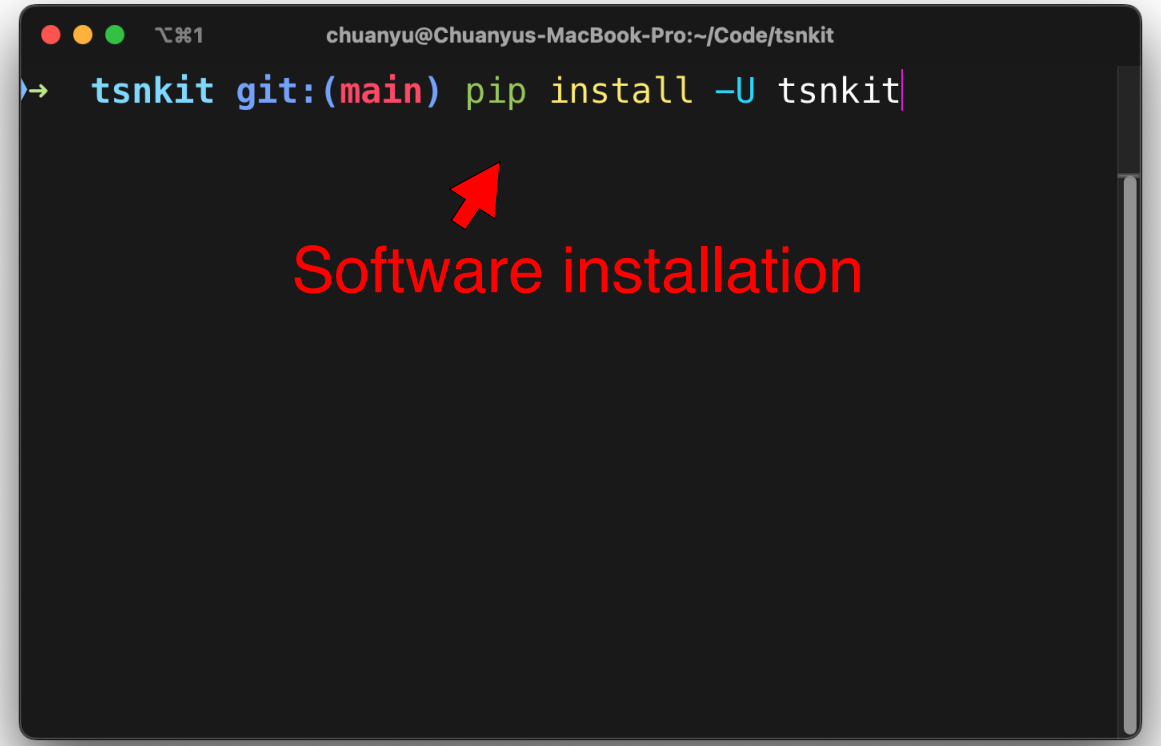
Use a wide range of experimental settings to fairly evaluate the strengths and limitations of different methods



Reproducibility

An open-source toolkit is available for the community

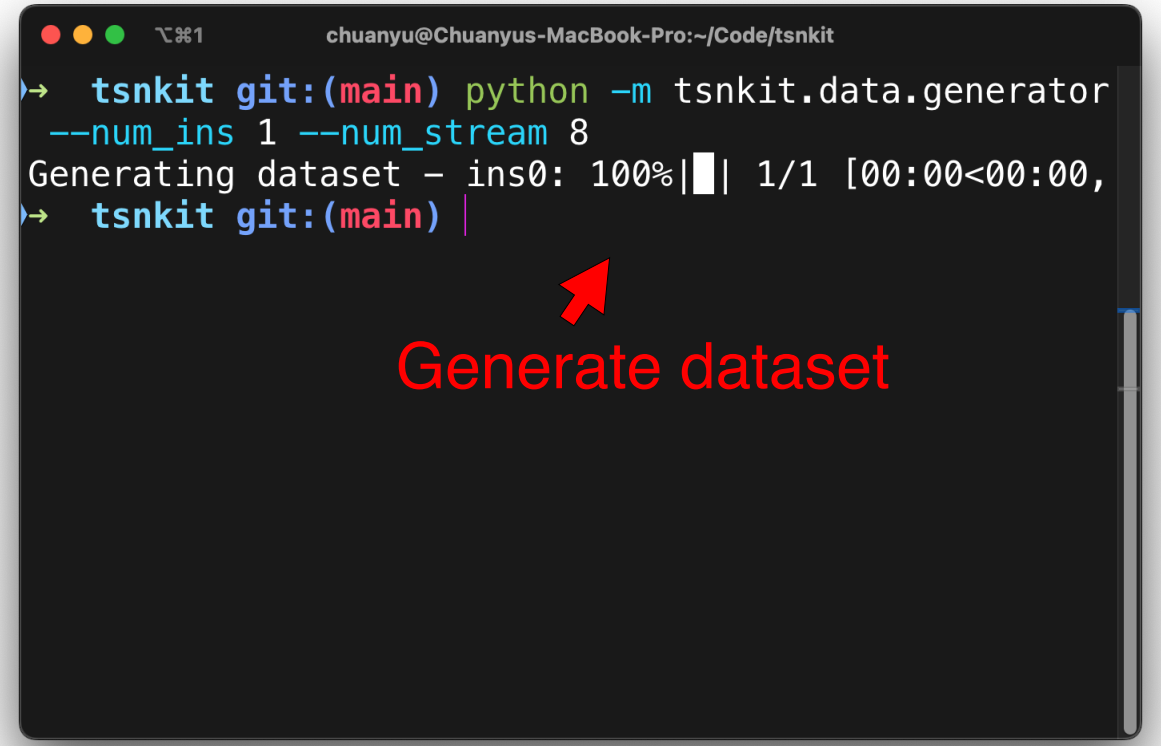
- Available at: `github.com/ChuanyuXue/tsnkit`
  - 32 Stars, 9 Forks, 6 issues closed
- Easy to install and use for future research



Reproducibility

An open-source toolkit is available for the community

- Available at: `github.com/ChuanyuXue/tsnkit`
  - 32 Stars, 9 Forks, 6 issues closed
- Easy to install and use for future research



Reproducibility

An open-source toolkit is available for the community

- Available at: `github.com/ChuanyuXue/tsnkit`
  - 32 Stars, 9 Forks, 6 issues closed
- Easy to install and use for future research

```
chuanyu@Chuanyus-MacBook-Pro:~/Code/tsnkit
➜  tsnkit git:(main) python -m tsnkit.data.generator
 --num_ins 1 --num_stream 8
Generating dataset - ins0: 100%|▉| 1/1 [00:00<00:00,
➜  tsnkit git:(main) cat dataset_logs.csv
id,size,period,deadline,topo,num_stream,num_sw
1,2,1,1,0,8,8
➜  tsnkit git:(main) python -m tsnkit.models.ls 1_ta
sk.csv 1_topo.csv
| time          | name          | flag    | solve_tim
e | total_time | total_mem
| 06~22:14:56   | -             | succ    | 0.004
  | 1.058      | 86400.0
➜  tsnkit git:(main)
```

Calculate schedule

Reproducibility

An open-source toolkit is available for the community

- Available at: `github.com/ChuanyuXue/tsnkit`
  - 32 Stars, 9 Forks, 6 issues closed
- Easy to install and use for future research

```
chuanyu@Chuanyus-MacBook-Pro:~/Code/tsnkit
→  tsnkit git:(main) python -m tsnkit.simulation.tas
 ./1_task.csv ./
  0%|                    | 0/200000 [00:00<?, ?it/s]
[Bridge (10, 2)]:   Flow 0 - Trans at 0
[Bridge (11, 3)]:   Flow 6 - Trans at 0
[Bridge (12, 4)]:   Flow 7 - Trans at 0
[Bridge (15, 7)]:   Flow 3 - Trans at 0
[Bridge (9, 1)]:    Flow 5 - Trans at 0
[Talker 10]:        Flow 0 - Send at 2800
[Bridge (2, 1)]:    Flow 0 - Arrive at 2800
[Bridge (2, 1)]:    Flow 0 - Trans at 2800
[Talker 12]:        Flow 7 - Send at 3600
[Bridge (4, 3)]:    Flow 7 - Arrive at 3600
[Bridge (4, 3)]:    Flow 7 - Trans at 3600
[Talker 11]:        Flow 6 - Send at 4400
[Bridge (3, 4)]:    Flow 6 - Arrive at 4400
```

Run simulation

Reproducibility

An open-source toolkit is available for the community

- Available at: `github.com/ChuanyuXue/tsnkit`
  - 32 Stars, 9 Forks, 6 issues closed
- Easy to install and use for future research

Testbed Setup

Motivation: The correctness of scheduling methods/model has not been validated on real hardware.

- Calibrate model parameters (e.g., delays, synchronization error)
- Validate the correctness of each scheduling method

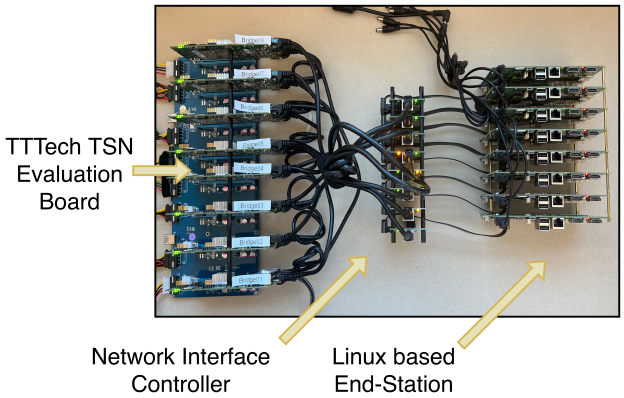
Testbed Setup

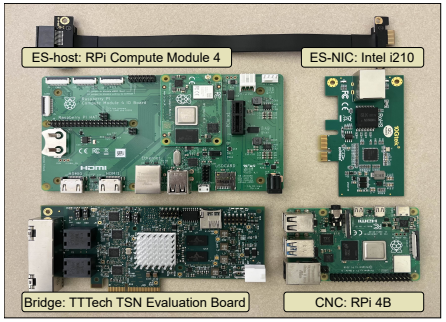
Motivation: The correctness of scheduling methods/model has not been validated on real hardware.

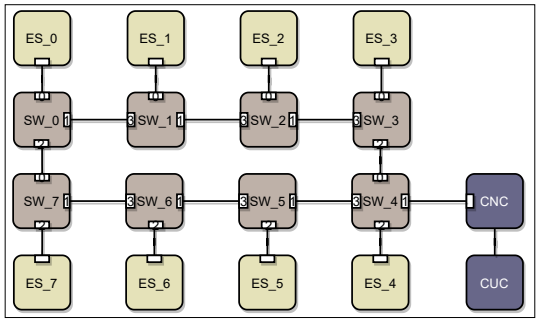
- Calibrate model parameters (e.g., delays, synchronization error)
- Validate the correctness of each scheduling method

Components:

- 8 Bridges: FPGA hardware-based TTTech TSN evaluation board
- 8 End-stations: Intel i210 NIC + Linux Ethernet stack
- Connected into a ring topology





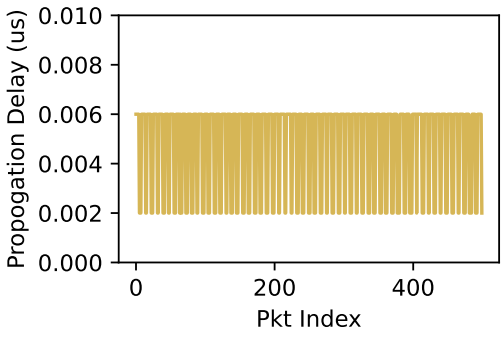


Testbed Validation

Motivation: Incorrect model assumptions may generate a feasible schedule but invalid on real hardware

Solution: Measurements of delays and synchronization error

- Propagation delay: $\leq$ 6 ns

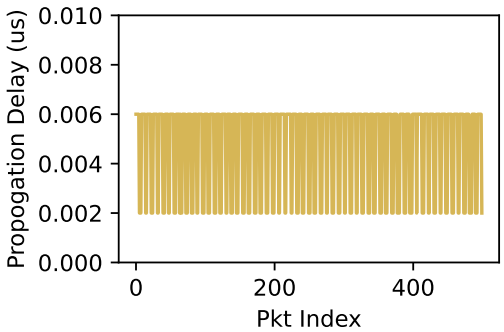


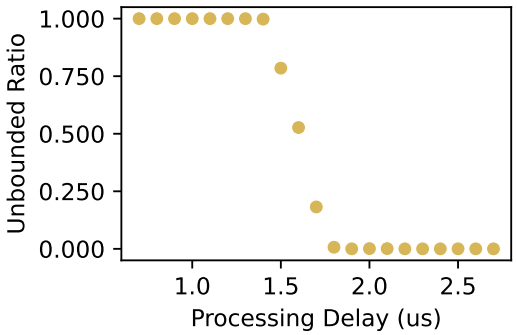
Testbed Validation

Motivation: Incorrect model assumptions may generate a feasible schedule but invalid on real hardware

Solution: Measurements of delays and synchronization error

- Propagation delay: $\leq$ 6 ns
- Processing delay: $\leq$ 1.9 μs



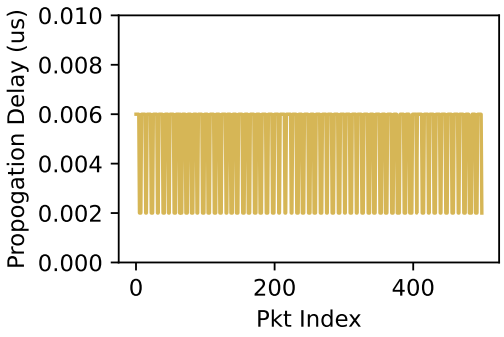


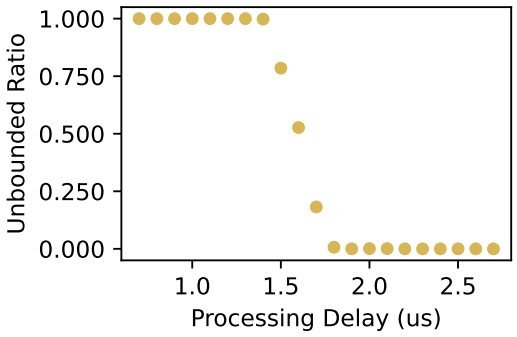
Testbed Validation

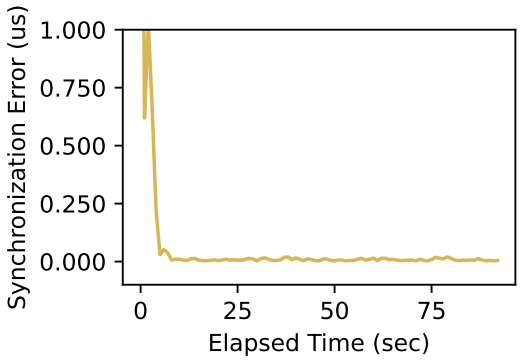
Motivation: Incorrect model assumptions may generate a feasible schedule but invalid on real hardware

Solution: Measurements of delays and synchronization error

- Propagation delay: $\leq$ 6 ns
- Processing delay: $\leq$ 1.9 μs
- Synchronization error: $\leq$ 10 ns (after 5 secs)



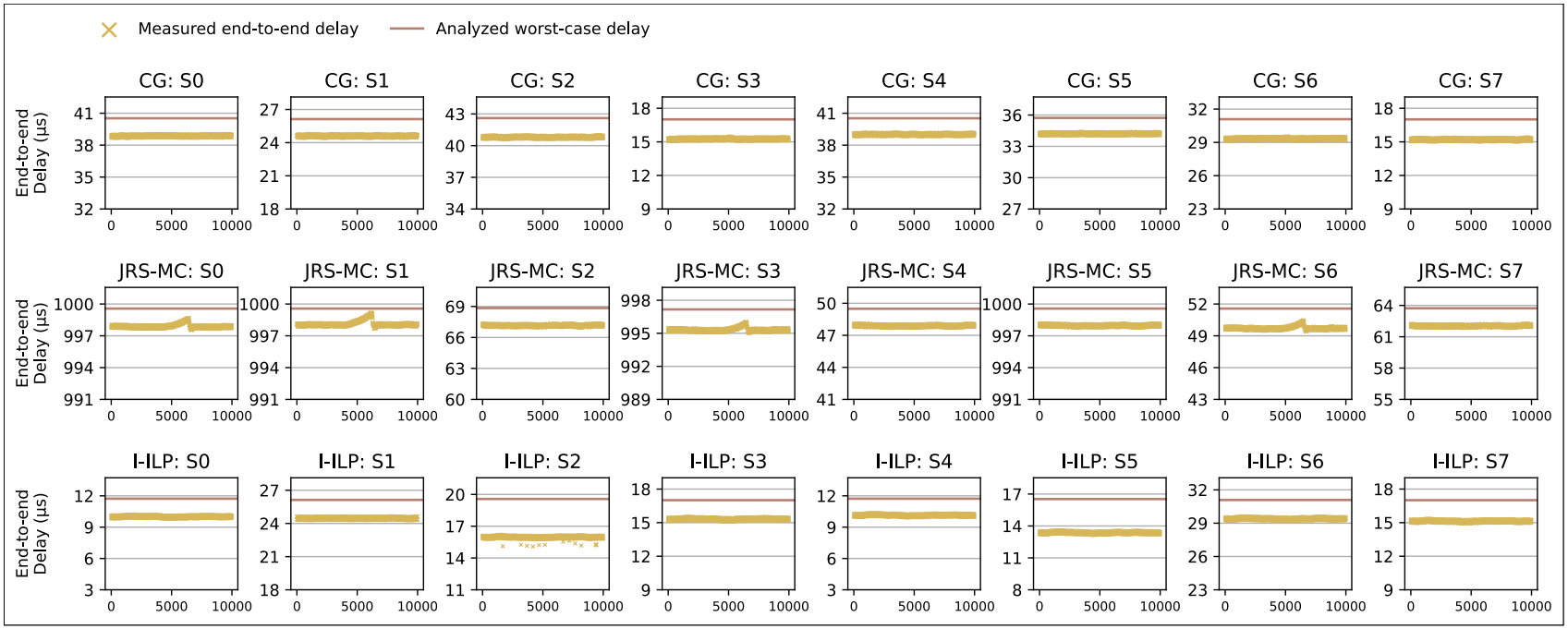




Testbed Validation

Performance validation of scheduling methods.

- Measured end-to-end delay vs. calculated worst-case delay
- Delay for reproduced methods are validated
- Jitter is observed due to collision between sync and TT traffic



Conclusion and Future work

Conclusion:

- Systematically categorization of TAS-based real-time scheduling models
- Quantitatively comparisons among 17 representative TAS-based real-time scheduling methods in TSN
- Real-world testbed for the performance validation of existing scheduling model and methods

Future Work:

- Incorporating fault tolerance scenarios and various traffic shapers
- More comprehensive empirical experiments on real-world testbed
- Encourage the community to utilize our open-source toolkit

Thank you

Q & A

Evaluation Metrics

Two evaluation scenarios:

- Scenario 1: Scheduleable ratio (SR). Assume unknown results as unschedulable
  - The ratio of schedulable instances to the total instances.
  - Example: $\mathrm{SR(A)} = 0.25 < \mathrm{SR(B)} = 0.5$
- Scenario 2: Schedulability advantage (SA). Pairwise performance comparison without considering unknown
  - $\Phi(A, B)$: The number of instances that A is schedulable but B is unschedulable.
  - $\Phi(A, B) > \Phi(B, A) = 0$: Method A dominates method B.
  - Example: $\Phi(A, B) = 1 > \Phi(B, A) = 0$

	Instance 1	Instance 2	Instance 3	Instance 4
Method A	✓	?	✗	?
Method B	✗	✓	?	✓