Dependence Graph Preprocessing for Faster Exact Modulo Scheduling in High-level Synthesis

Julian Oppermann, Melanie Reuter-Oppermann, Lukas Sommer, Oliver Sinnen, Andreas Koch

TECHNISCHE UNIVERSITÄT DARMSTADT

KIT Karlsruhe Institute of Technology

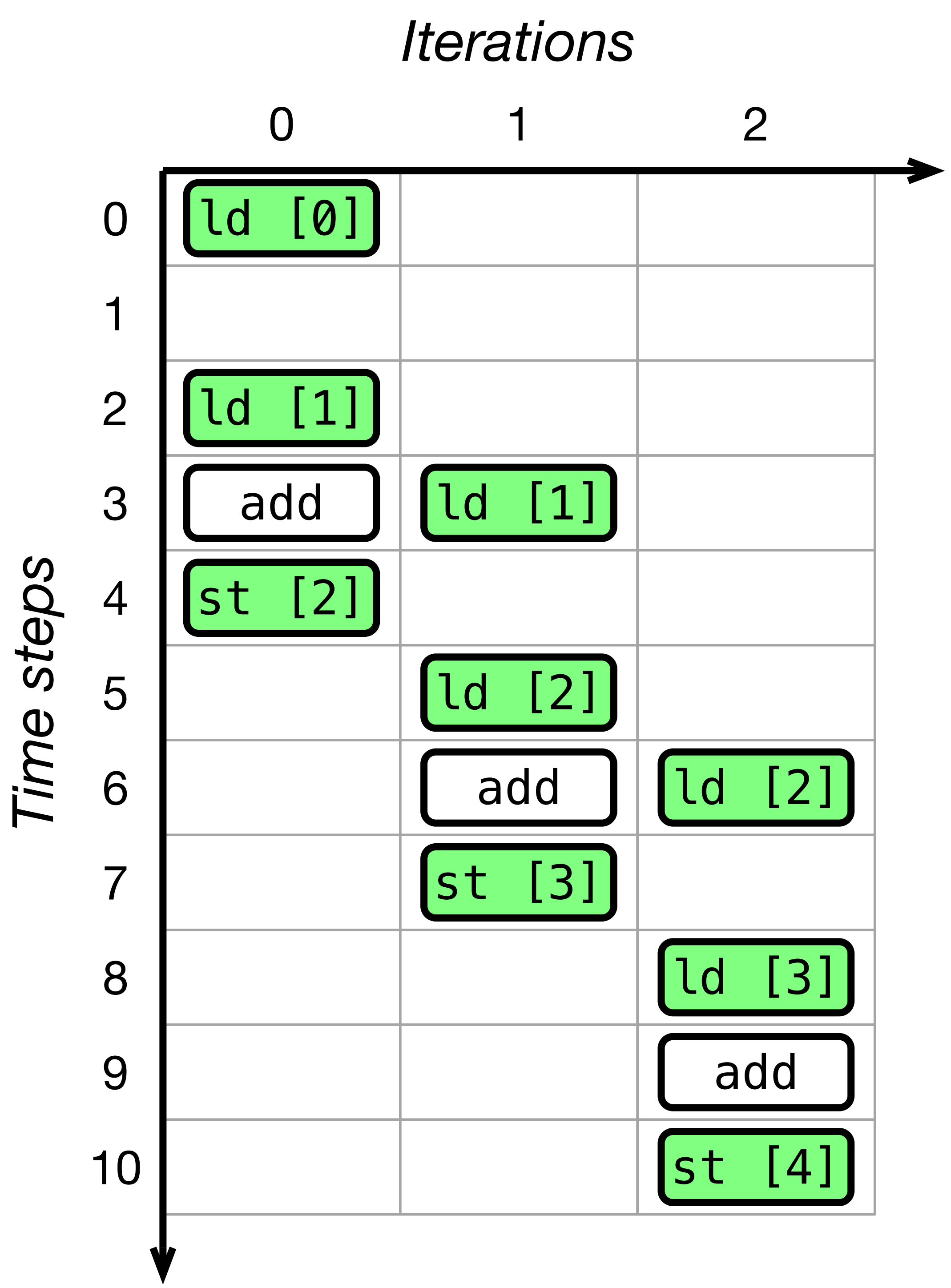
THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND

Agenda

- **(Short) introduction to modulo scheduling**
- Proposed preprocessing approach
- Results and insights

Modulo Scheduling

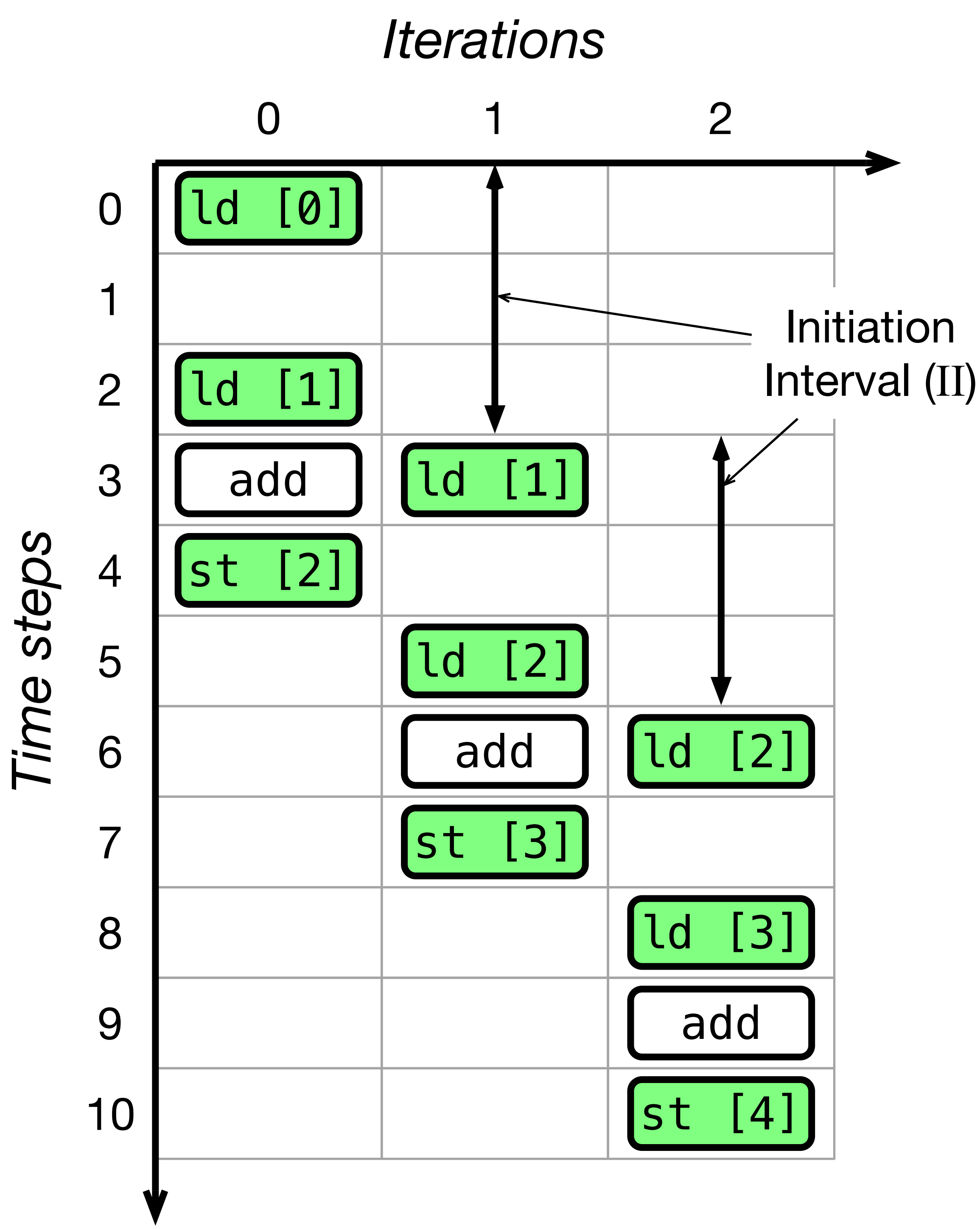
- **Loop pipelining**
  = increase throughput by overlapping iterations



Modulo Scheduling

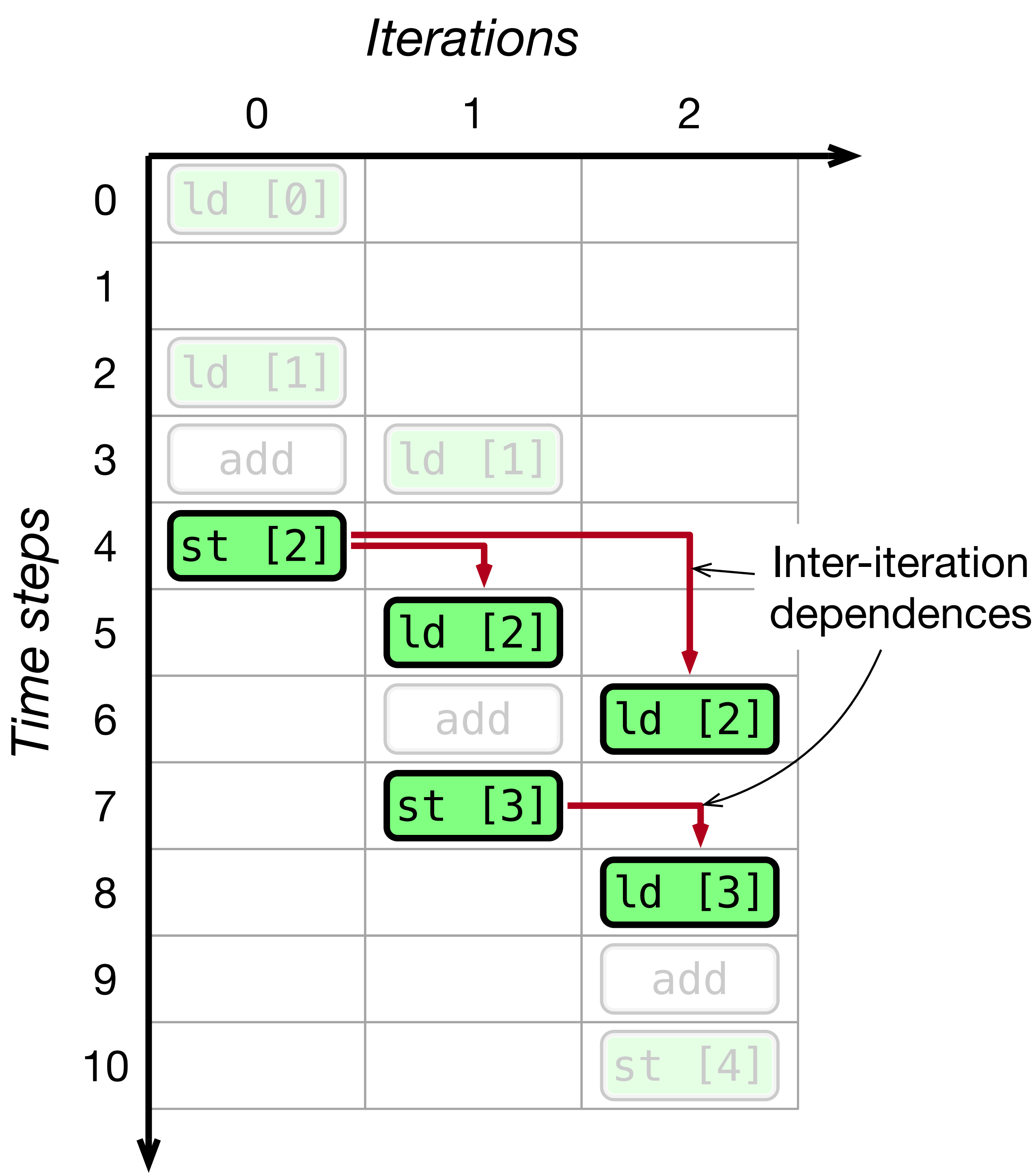
- **Loop pipelining**
  = increase throughput by overlapping iterations

- **Modulo schedulers** compute
  - initiation interval (II)
  - start times (= „schedule“)



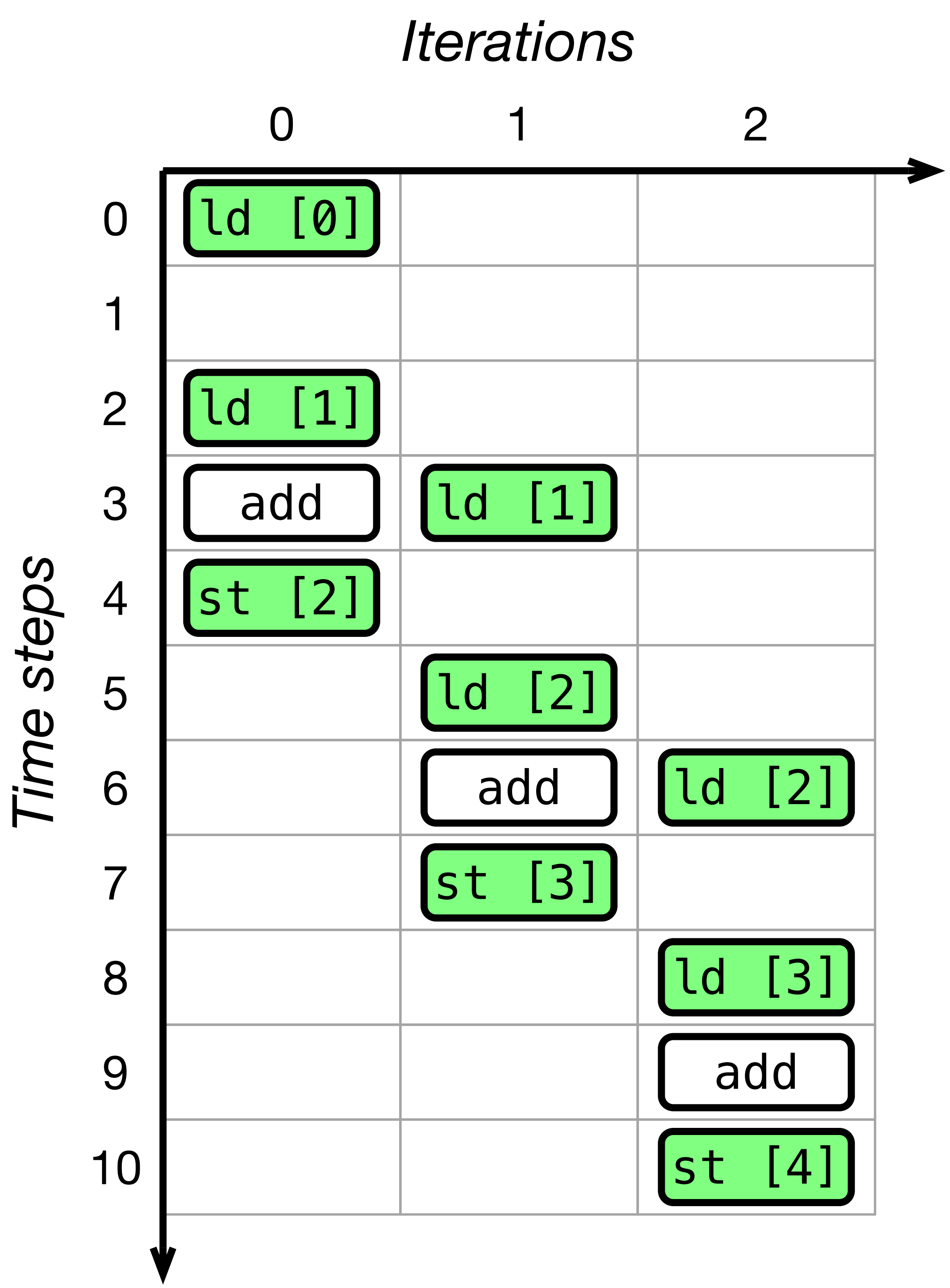
Modulo Scheduling

- **Loop pipelining**
  = increase throughput by overlapping iterations
- **Modulo schedulers** compute
  - initiation interval (II)
  - start times (= „schedule")
- Subject to
  - inter-iteration dependences



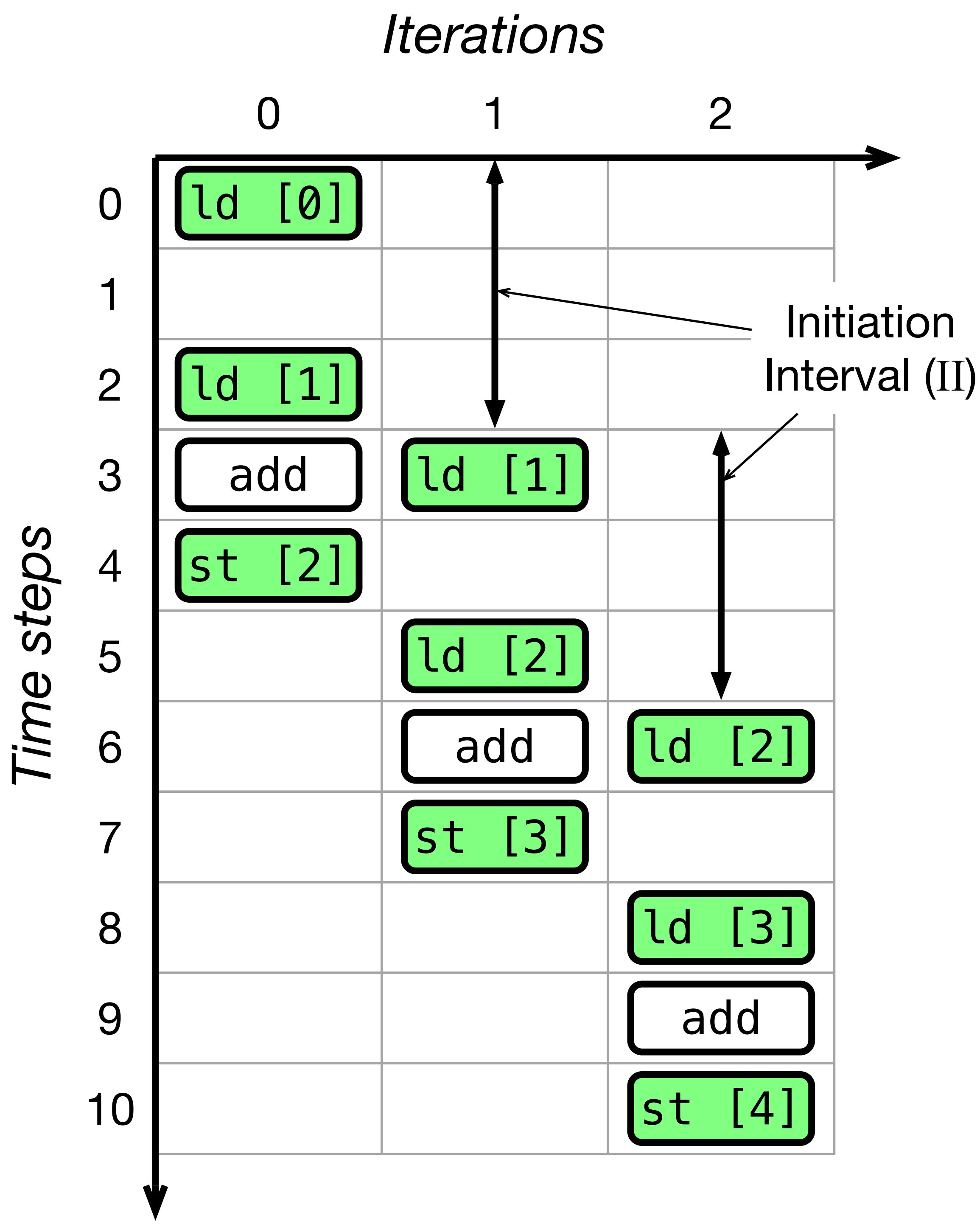
Modulo Scheduling

- **Loop pipelining**
  = increase throughput by overlapping iterations
- **Modulo schedulers** compute
  - initiation interval (II)
  - start times (= „schedule“)
- Subject to
  - inter-iteration dependences
  - resource constraints



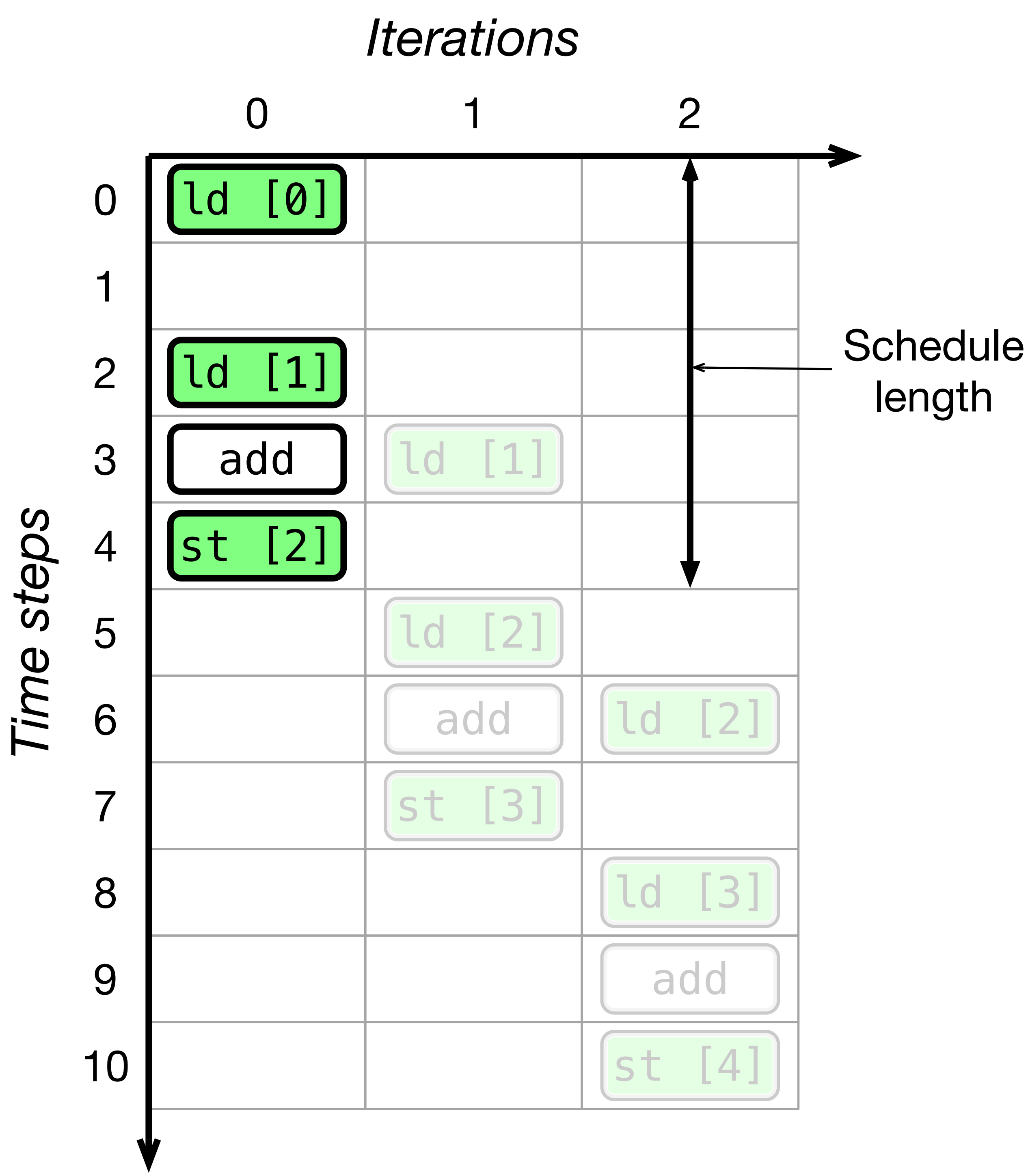
Modulo Scheduling

- **Loop pipelining**
  = increase throughput by overlapping iterations
- **Modulo schedulers** compute
  - initiation interval (II)
  - start times (= „schedule“)
- Subject to
  - inter-iteration dependences
  - resource constraints
- Minimise
  1. initiation interval



Modulo Scheduling

- **Loop pipelining**
  = increase throughput by overlapping iterations
- **Modulo schedulers** compute
  - initiation interval (II)
  - start times (= „schedule“)
- Subject to
  - inter-iteration dependences
  - resource constraints
- Minimise
  1. initiation interval
  2. schedule length



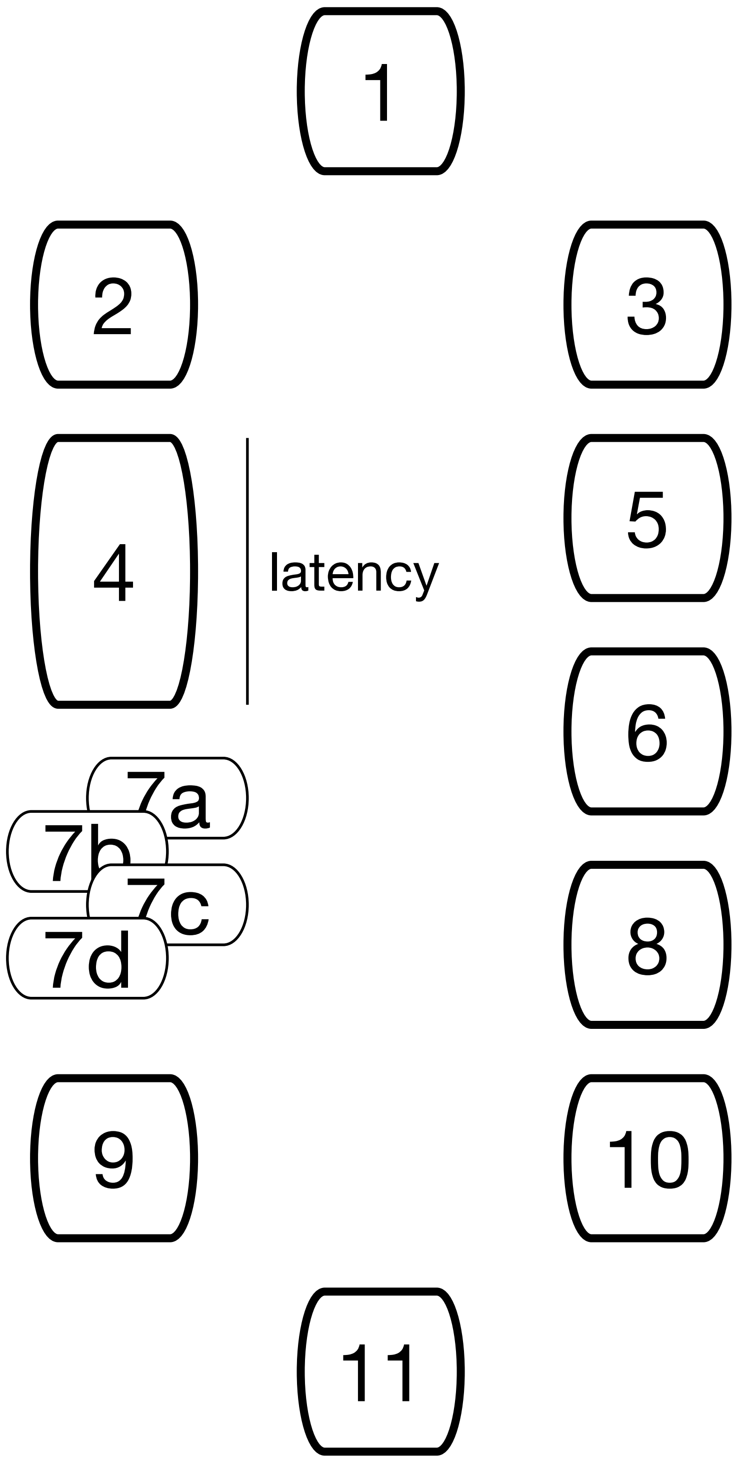
Formal Definition

An **instance** of the **modulo scheduling problem** (MSP) is defined by:

Formal Definition

An **instance** of the **modulo scheduling problem** (MSP) is defined by:

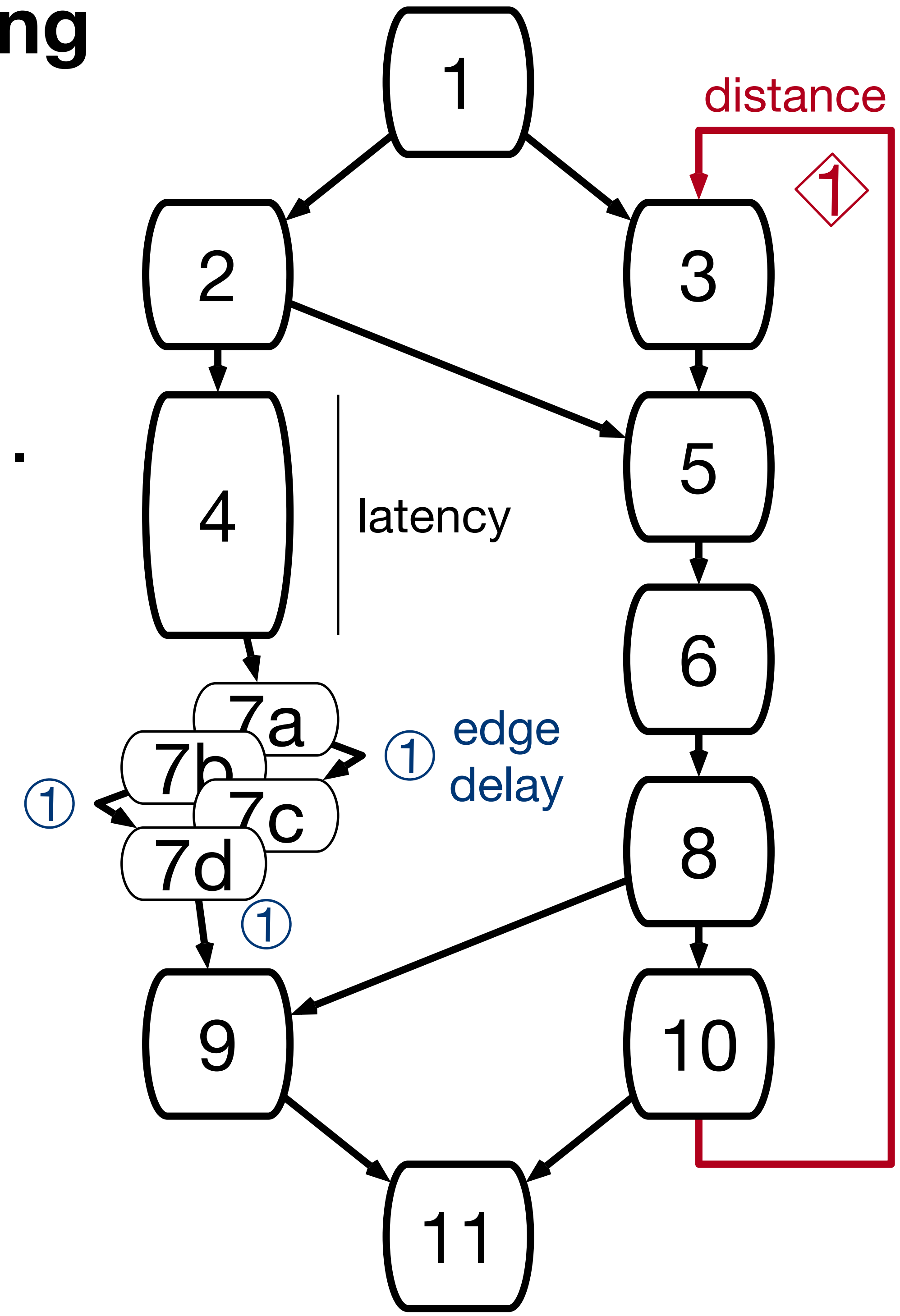
- Operations *i*
  - latency: 0 (combinatorial), 1, 2, …



Formal Definition

An **instance** of the **modulo scheduling problem** (MSP) is defined by:

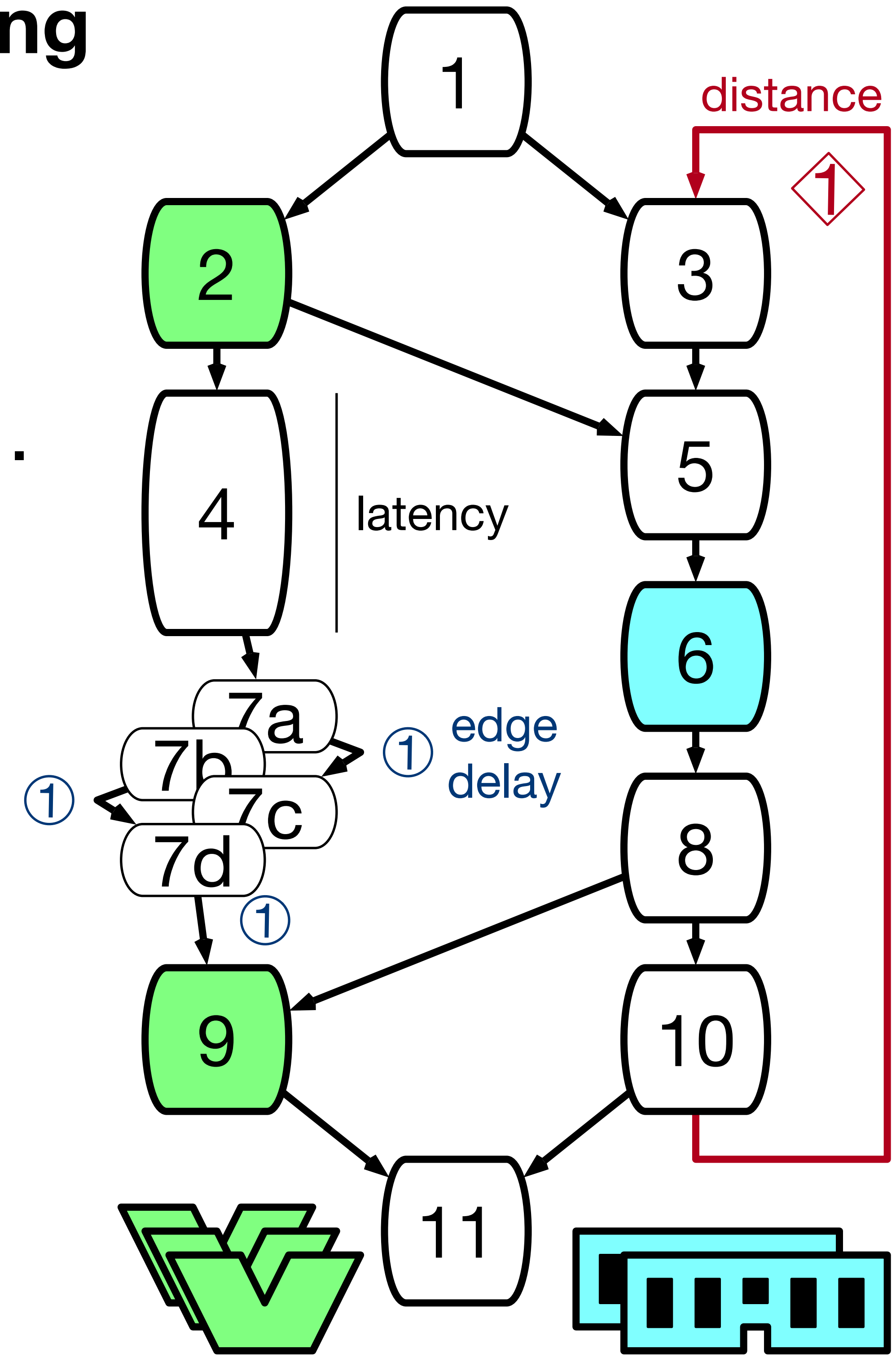
- Operations *i*
  - latency: 0 (combinatorial), 1, 2, …
- Edges $i \rightarrow j$
  - (delay), e.g. to control chaining
  - <distance>, ≥ 1 for inter-iteration dependences („backedges“)



Formal Definition

An **instance** of the **modulo scheduling problem** (MSP) is defined by:

- Operations *i*
  - latency: 0 (combinatorial), 1, 2, …
- Edges $i \rightarrow j$
  - (delay), e.g. to control chaining
  - <distance>, ≥ 1 for inter-iteration dependences („backedges“)
- Resource model
  - distinct types with given #units
  - unlimited operations



Motivation

- Modulo scheduling is usually applied to:

Motivation

- Modulo scheduling is usually applied to:
  - Very-long-instruction-word (**VLIW**) architectures
    - majority of literature targets VLIW compilers

Motivation

- Modulo scheduling is usually applied to:
  - Very-long-instruction-word (**VLIW**) architectures
    - majority of literature targets VLIW compilers
  - High-level synthesis (**HLS**)
    - **larger** and **denser** dependence graphs
    - **not all** operations are **resource-constrained**

Motivation

- Modulo scheduling is usually applied to:
  - Very-long-instruction-word (**VLIW**) architectures
    - majority of literature targets VLIW compilers
  - High-level synthesis (**HLS**)
    - **larger** and **denser** dependence graphs
    - **not all** operations are **resource-constrained**
- Observed scalability issues in highly-tuned approach

→ *Can't we just simplify the problem?*

Benchmark Set

- **21** medium to large MSP instances
  - from **CHStone** and **MachSuite**
  - extracted from a **typical HLS flow**

Benchmark Set

- **21** medium to large MSP instances
  - from **CHStone** and **MachSuite**
  - extracted from a **typical HLS flow**
- Median values
  - 471 operations

Benchmark Set

- **21** medium to large MSP instances
  - from **CHStone** and **MachSuite**
  - extracted from a **typical HLS flow**
- Median values
  - 471 operations
    - 12% resource-limited operations

Benchmark Set

- **21** medium to large MSP instances
  - from **CHStone** and **MachSuite**
  - extracted from a **typical HLS flow**
- Median values
  - 471 operations
    - 12% resource-limited operations
  - 1578 edges

Benchmark Set

- **21** medium to large MSP instances
  - from **CHStone** and **MachSuite**
  - extracted from a **typical HLS flow**

- Median values
  - 471 operations
    - 12% resource-limited operations
  - 1578 edges
    - 63% non-dataflow edges (memory dependences, chaining control)

Agenda

- ☑ (Short) introduction to modulo scheduling
- **Proposed preprocessing approach**
- ☐ Results and insights

Basic Insight

- Some operations are **critical** for the scheduling result, as they influence the
  - **feasibility**, and/or the
  - **objective value** of the solution

Basic Insight

- Some operations are **critical** for the scheduling result, as they influence the
  - **feasibility**, and/or the
  - **objective value** of the solution
- Others can always be scheduled ASAP

Basic Insight

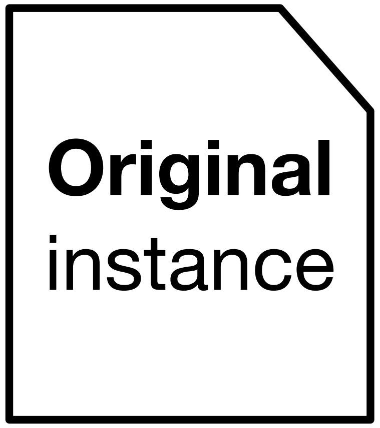
- Some operations are **critical** for the scheduling result, as they influence the
  - **feasibility**, and/or the
  - **objective value** of the solution
- Others can always be scheduled ASAP
  - find subgraphs of **non-critical** operations

Basic Insight

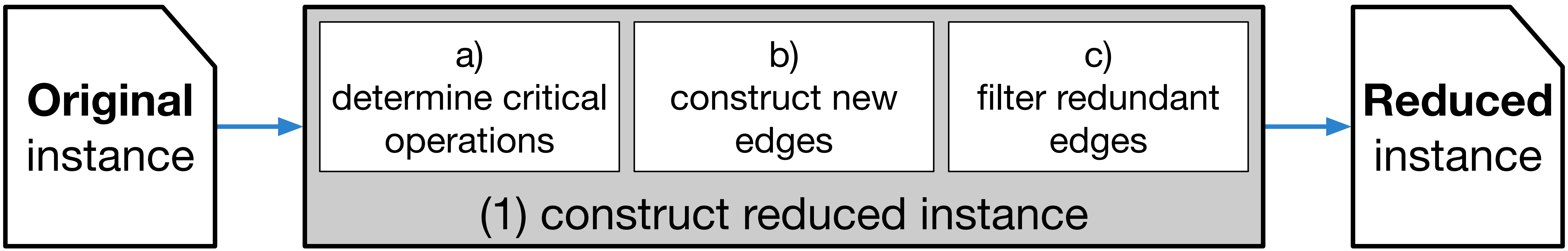
- Some operations are **critical** for the scheduling result, as they influence the
  - **feasibility**, and/or the
  - **objective value** of the solution

- Others can always be scheduled ASAP
  - find subgraphs of **non-critical** operations
  - replace by a single edge with appropriate delay

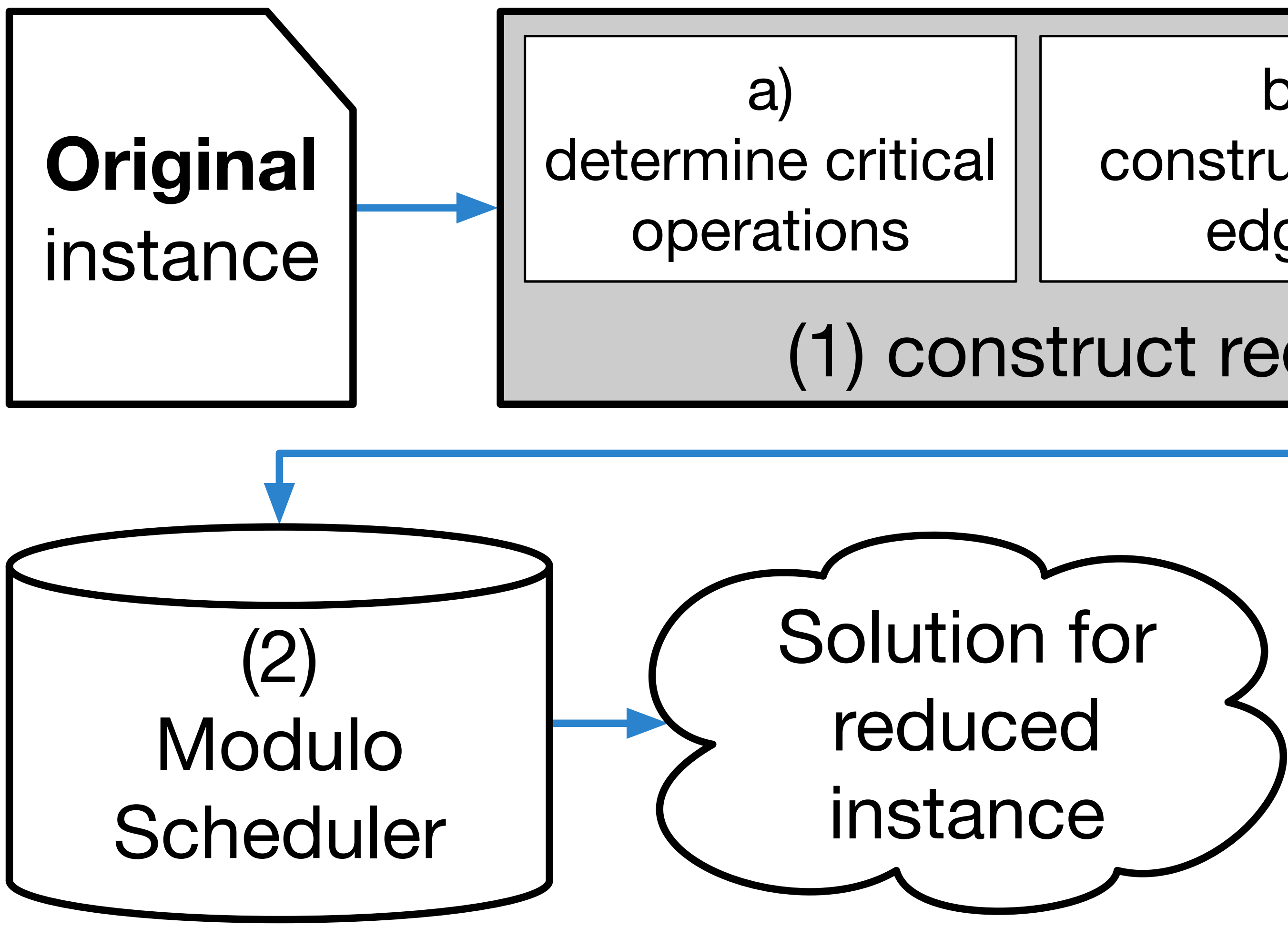
Approach Overview



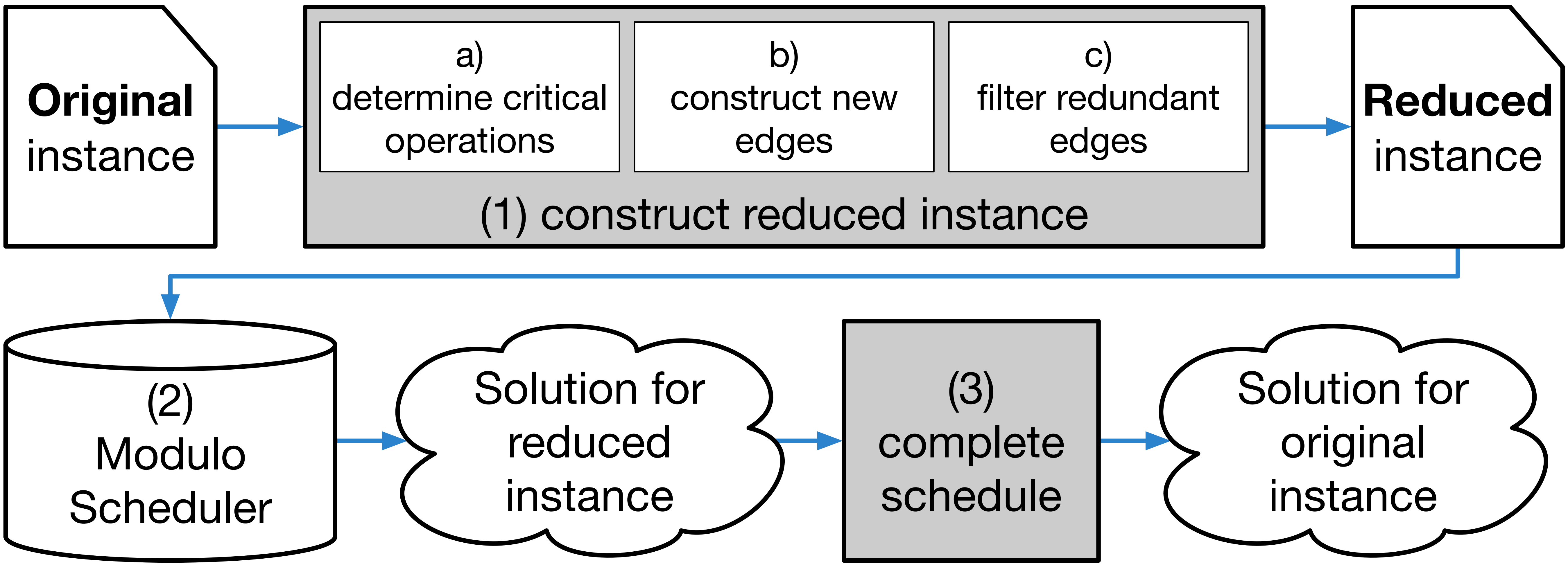
Approach Overview



Approach Overview

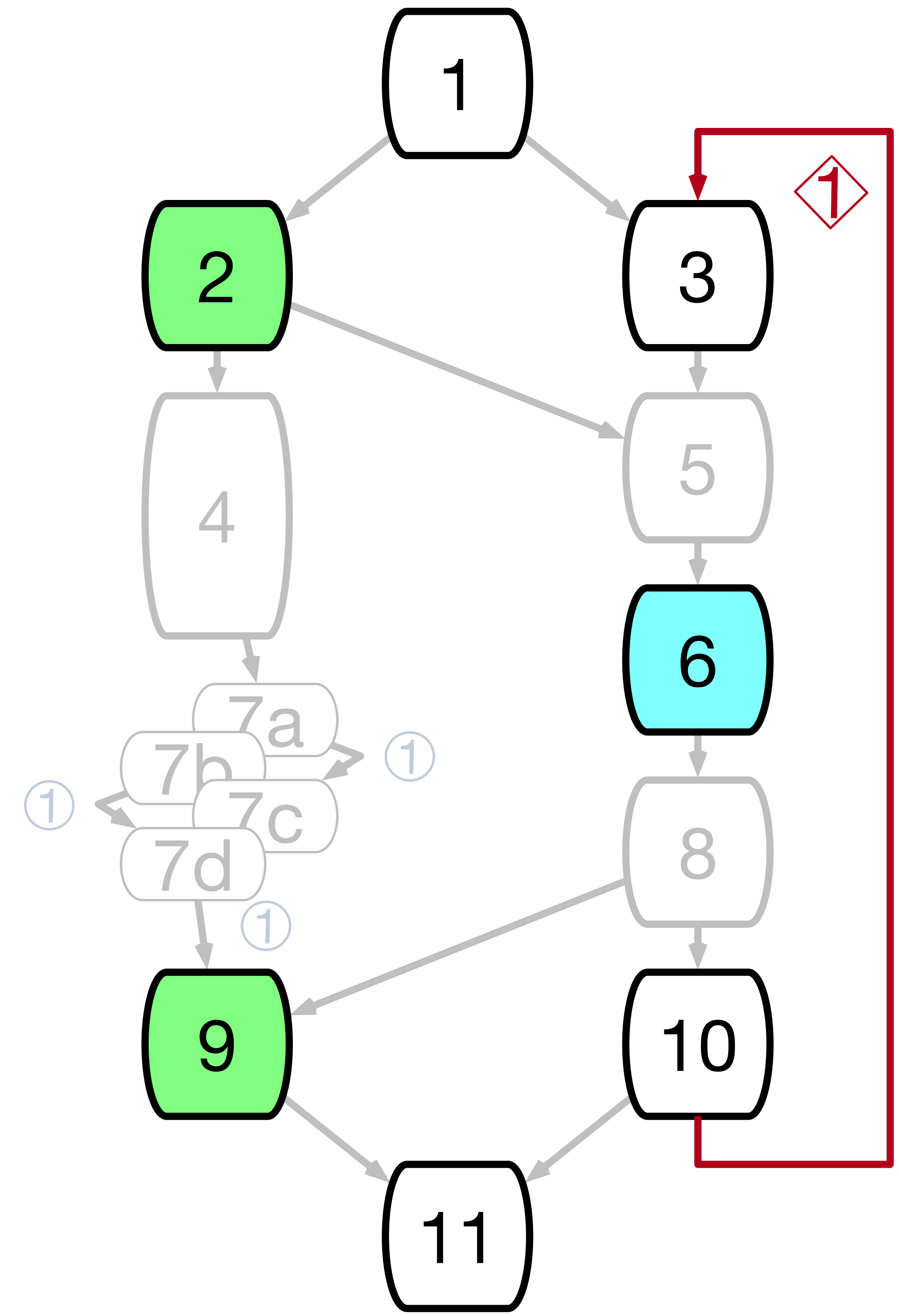


Approach Overview



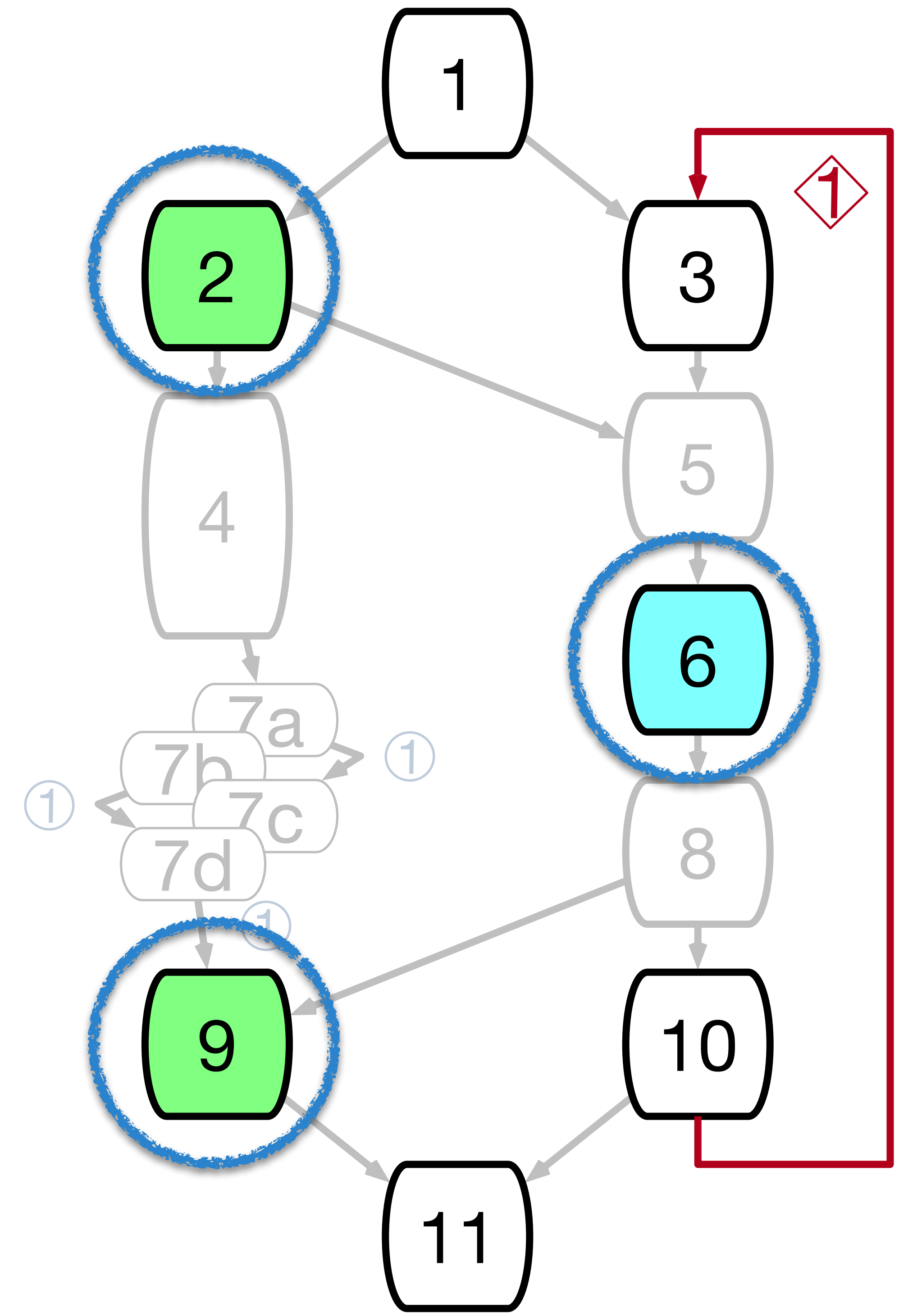
1a) Determine Critical Operations

- Feasibility



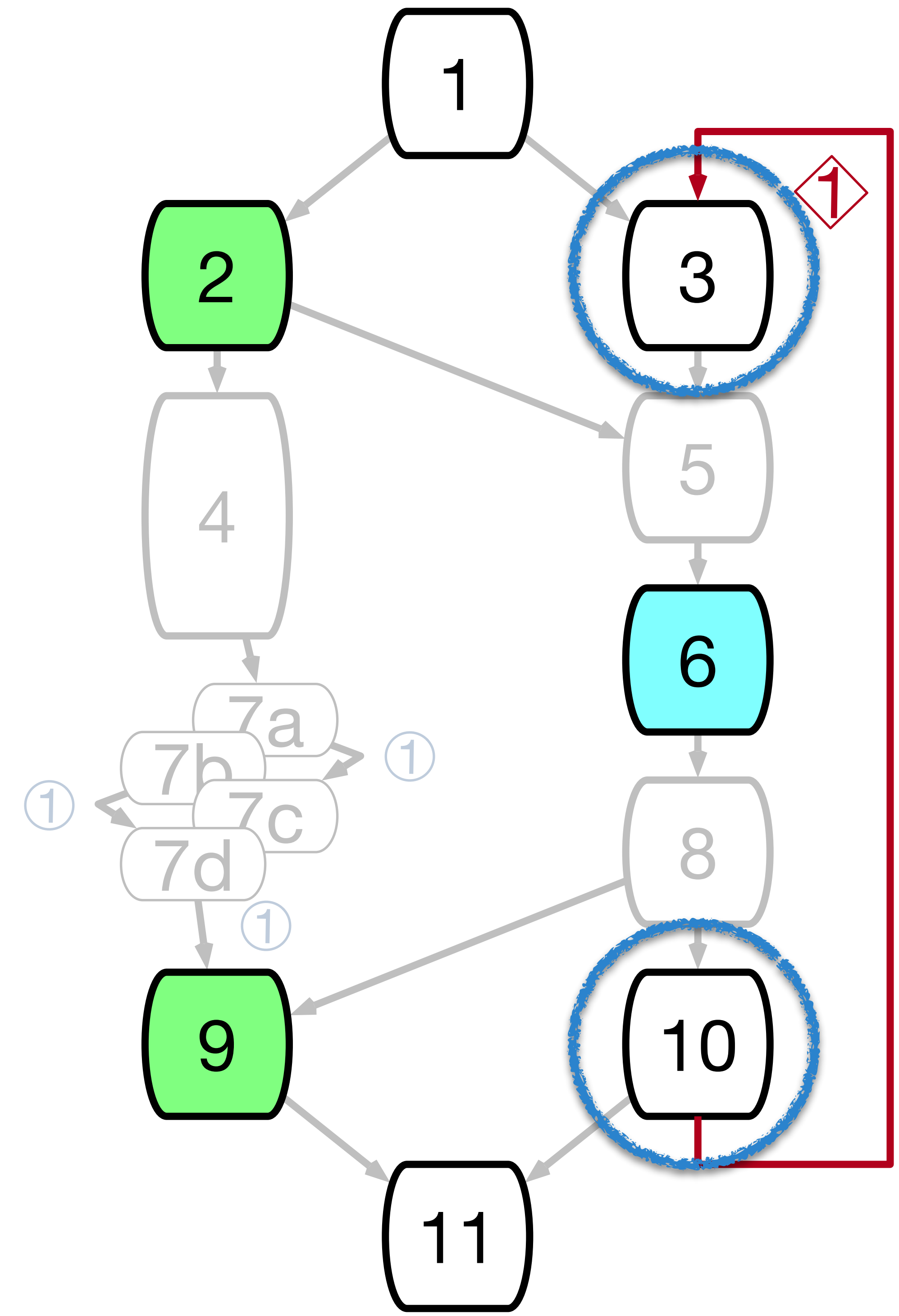
1a) Determine Critical Operations

- Feasibility
  - resource-limited operations



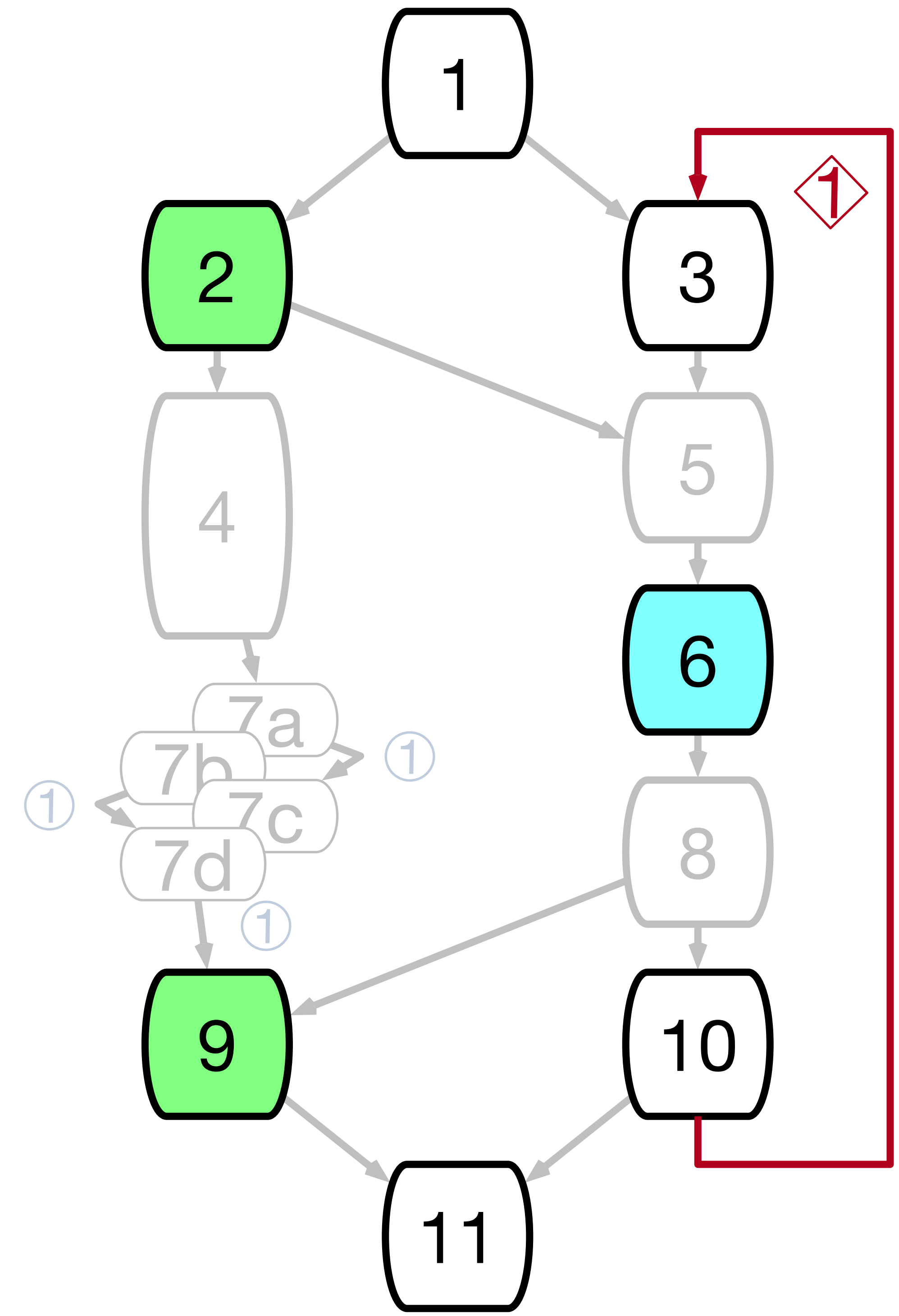
1a) Determine Critical Operations

- Feasibility
  - resource-limited operations
  - endpoints of backedges



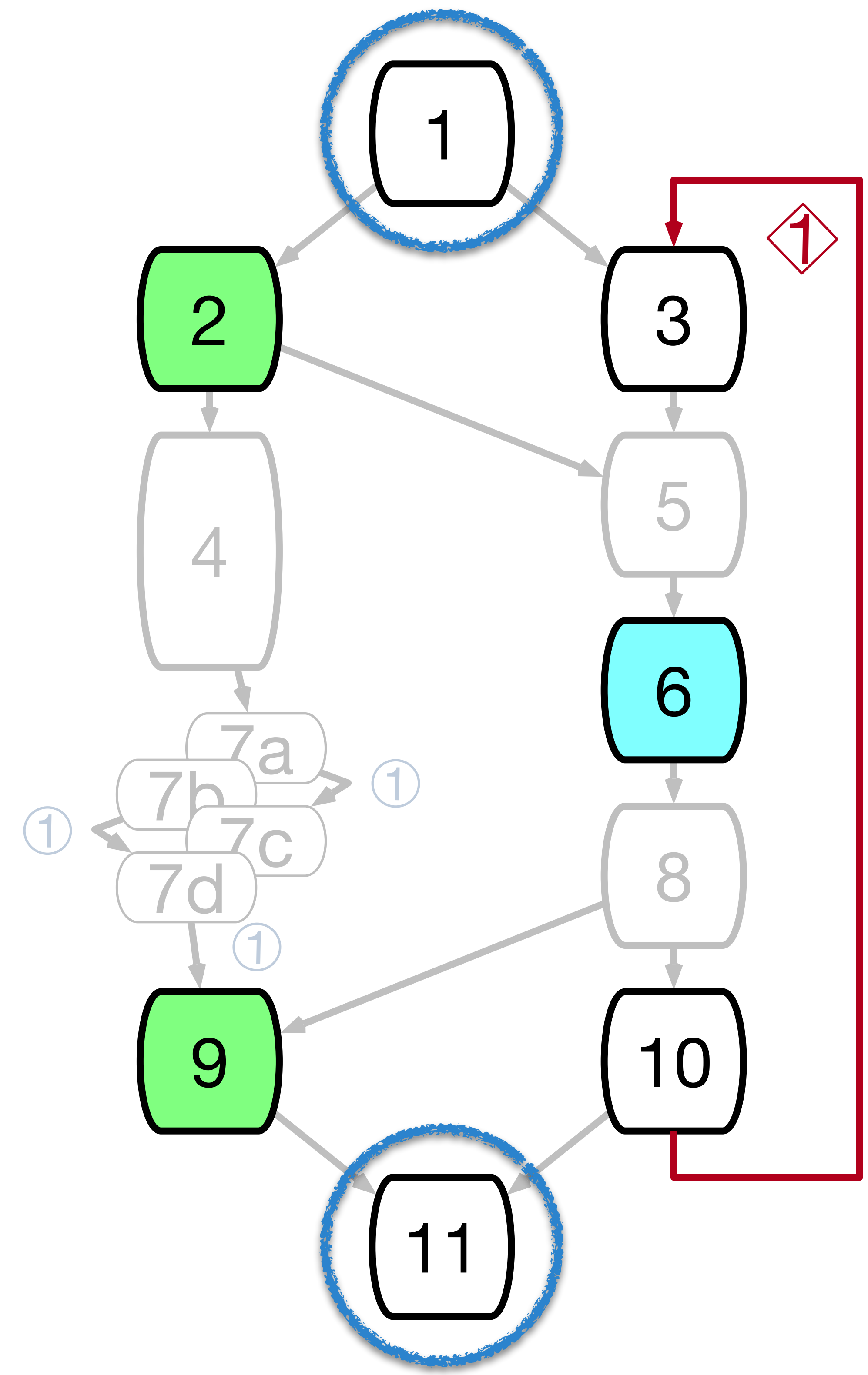
1a) Determine Critical Operations

- Feasibility
  - resource-limited operations
  - endpoints of backedges
- Objective



1a) Determine Critical Operations

- Feasibility
  - resource-limited operations
  - endpoints of backedges
- Objective
  - source and sink nodes



1b) Construct New Edges

- Single-pass data-flow analysis over dependence graph

1b) Construct New Edges

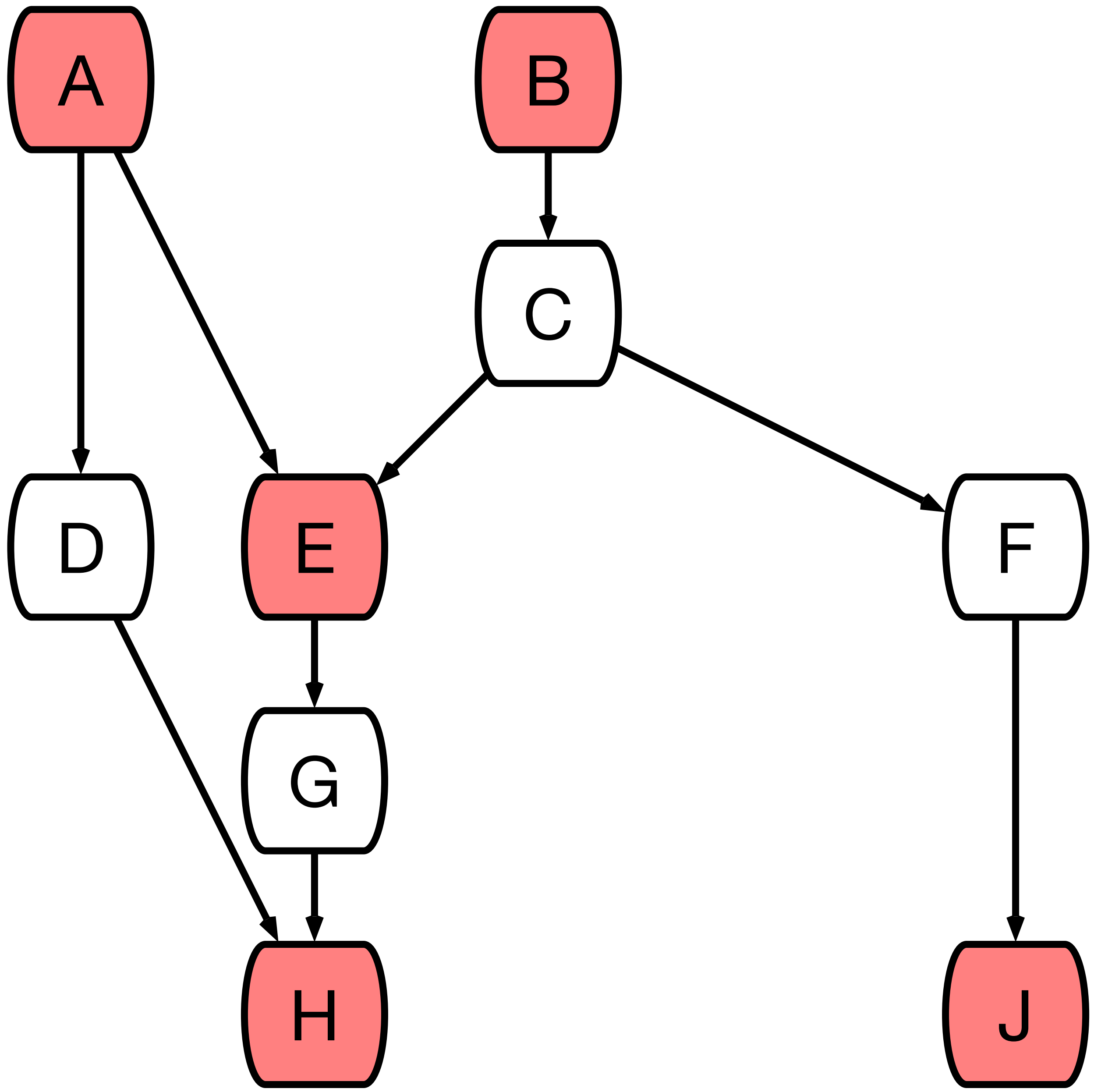
- Single-pass data-flow analysis over dependence graph
- For each operation, compute the **longest paths** to the **nearest** preceding critical operations
  - „length“ = accumulated latencies and delays

1b) Construct New Edges

- Single-pass data-flow analysis over dependence graph
- For each operation, compute the **longest paths** to the **nearest** preceding critical operations
  - „length“ = accumulated latencies and delays
  - paths are **reset** when encountering a critical operation

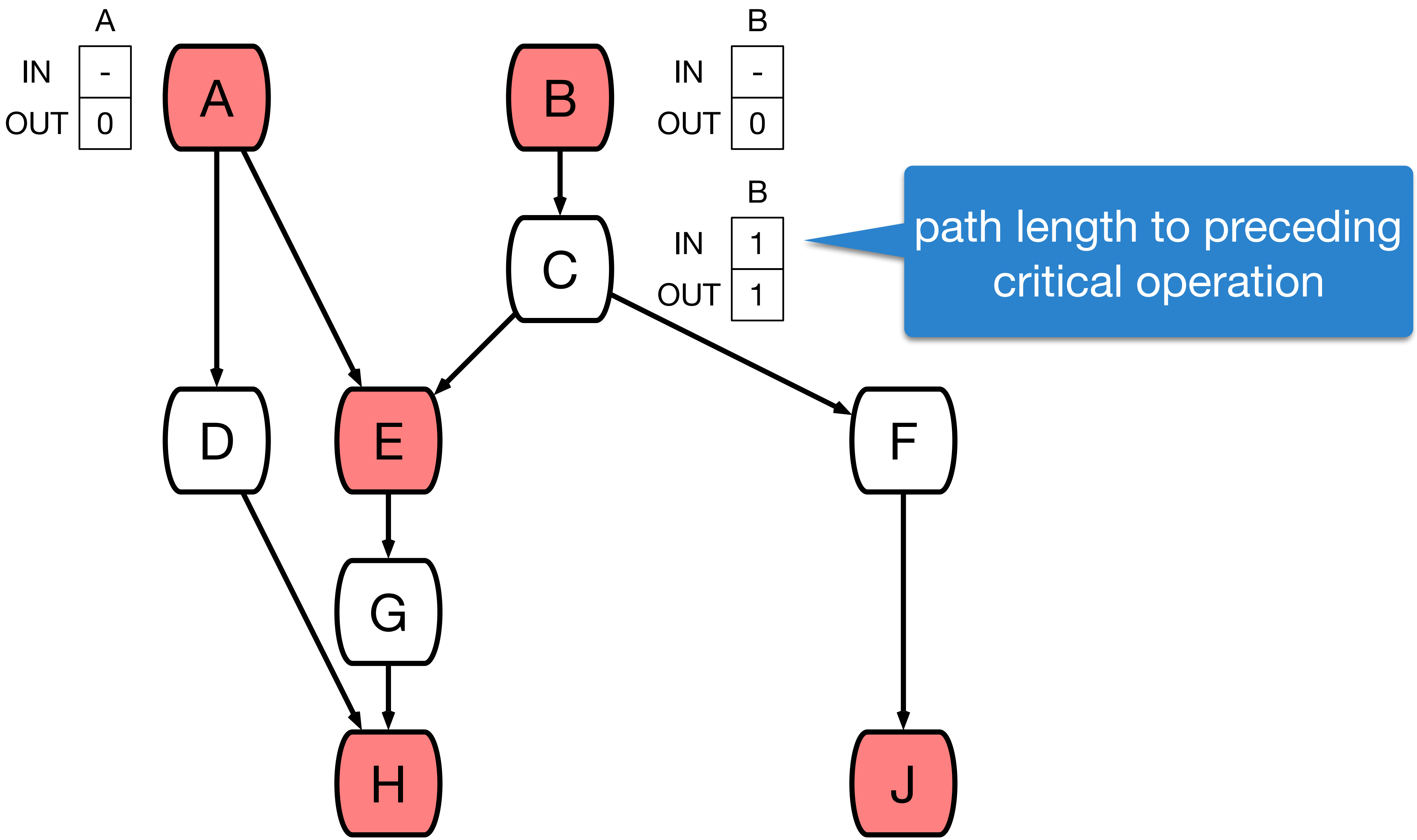
1b) Construct New Edges

- Example of data-flow analysis



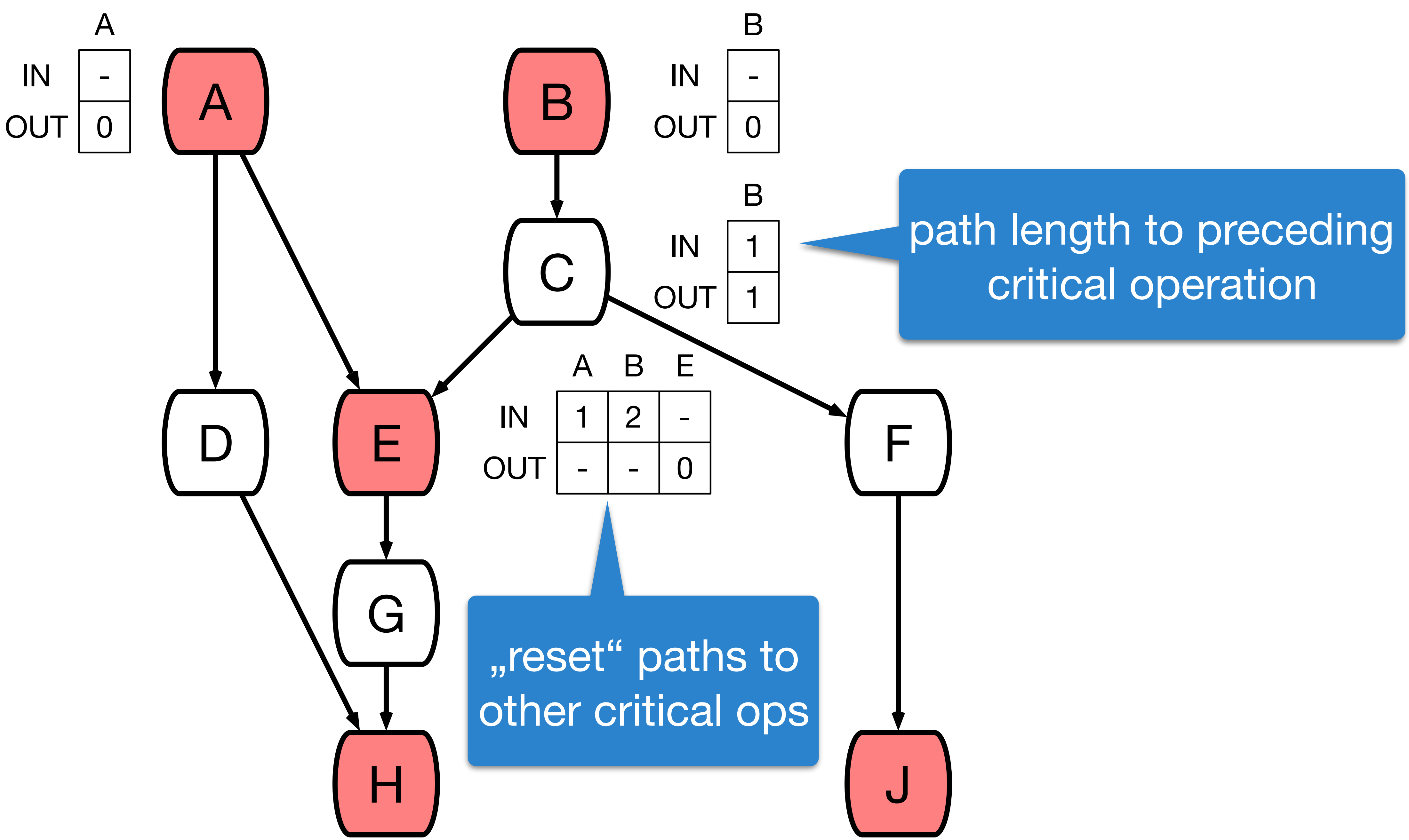
1b) Construct New Edges

- Example of data-flow analysis



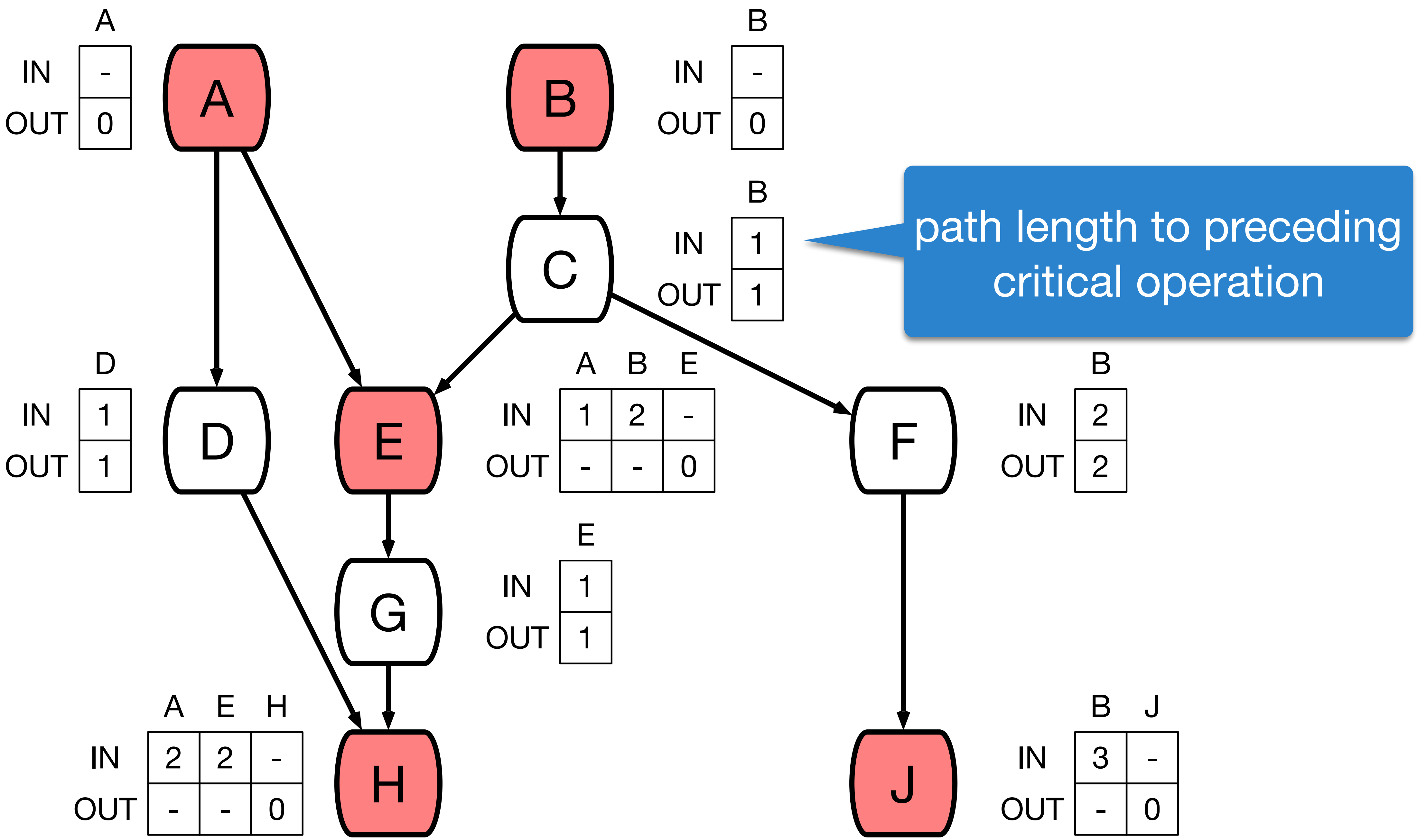
1b) Construct New Edges

- Example of data-flow analysis



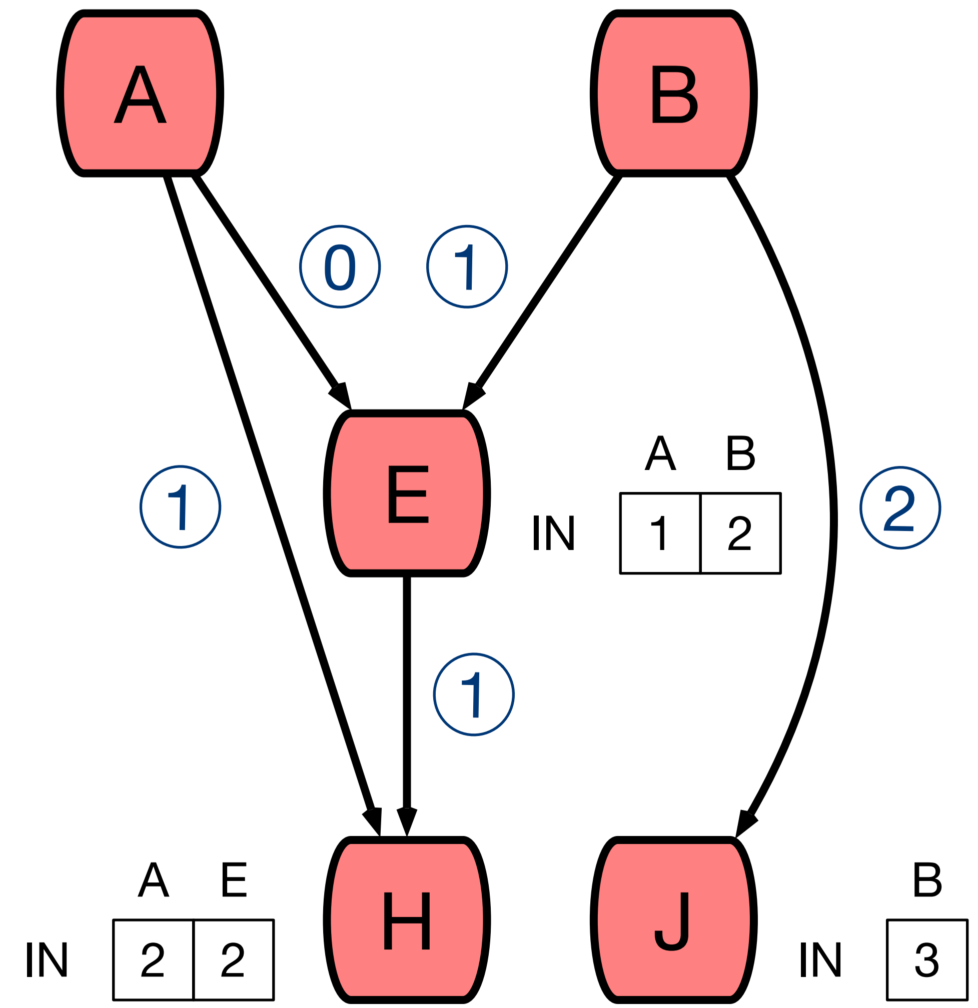
1b) Construct New Edges

- Example of data-flow analysis



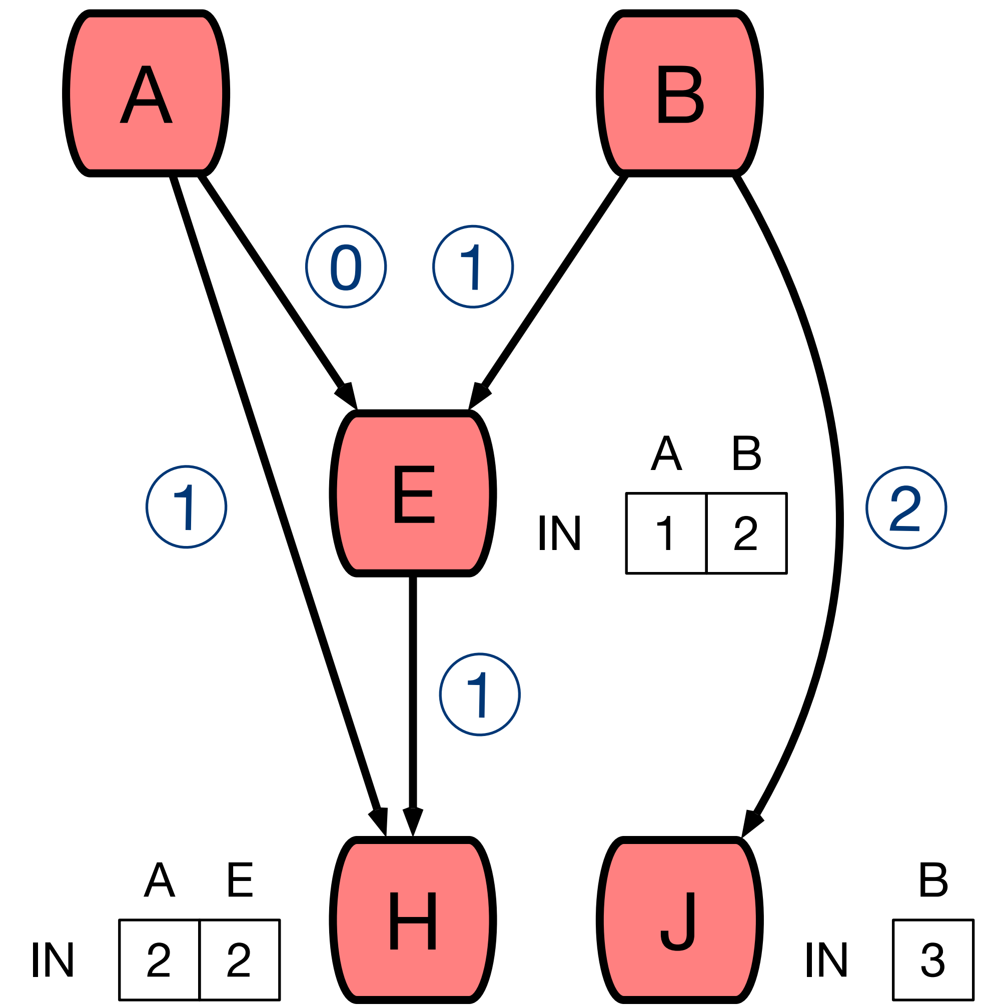
1b) Construct New Edges

- Construct edges between reachable critical operations



1b) Construct New Edges

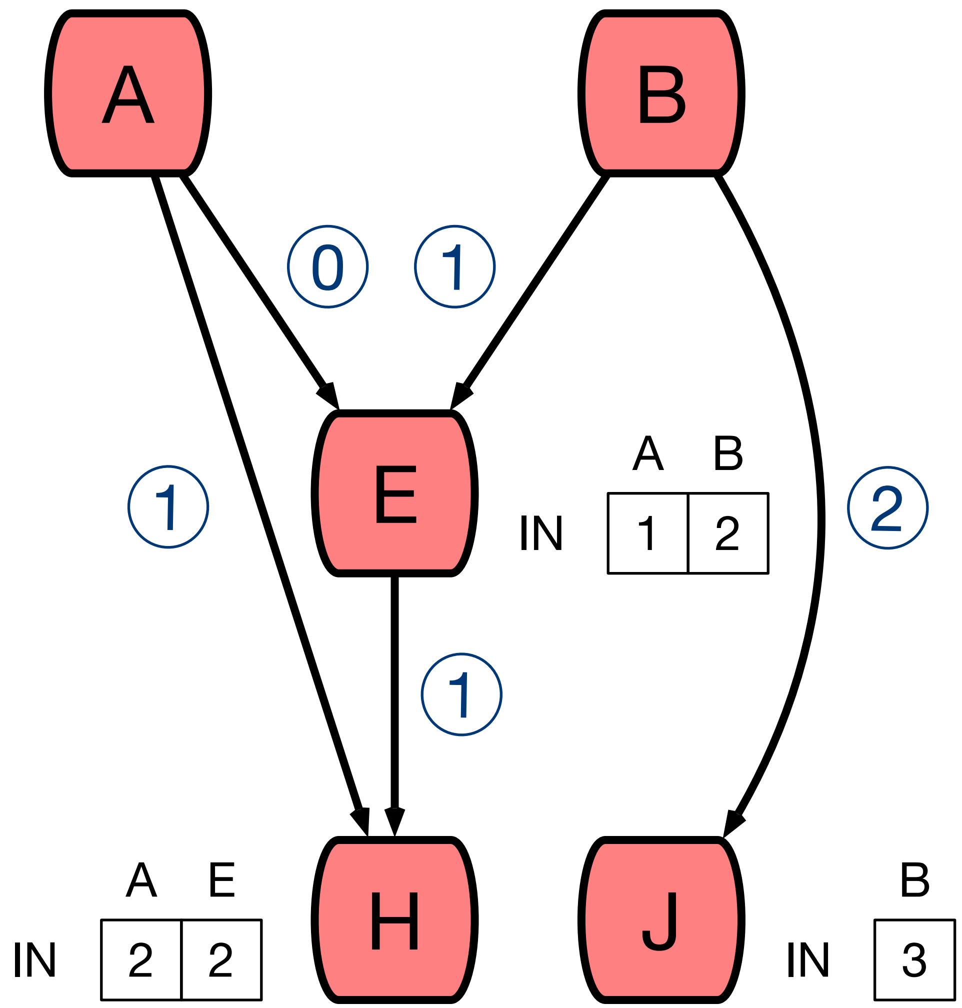
- Construct edges between reachable critical operations
  - need to subtract source operation's delay



1b) Construct New Edges

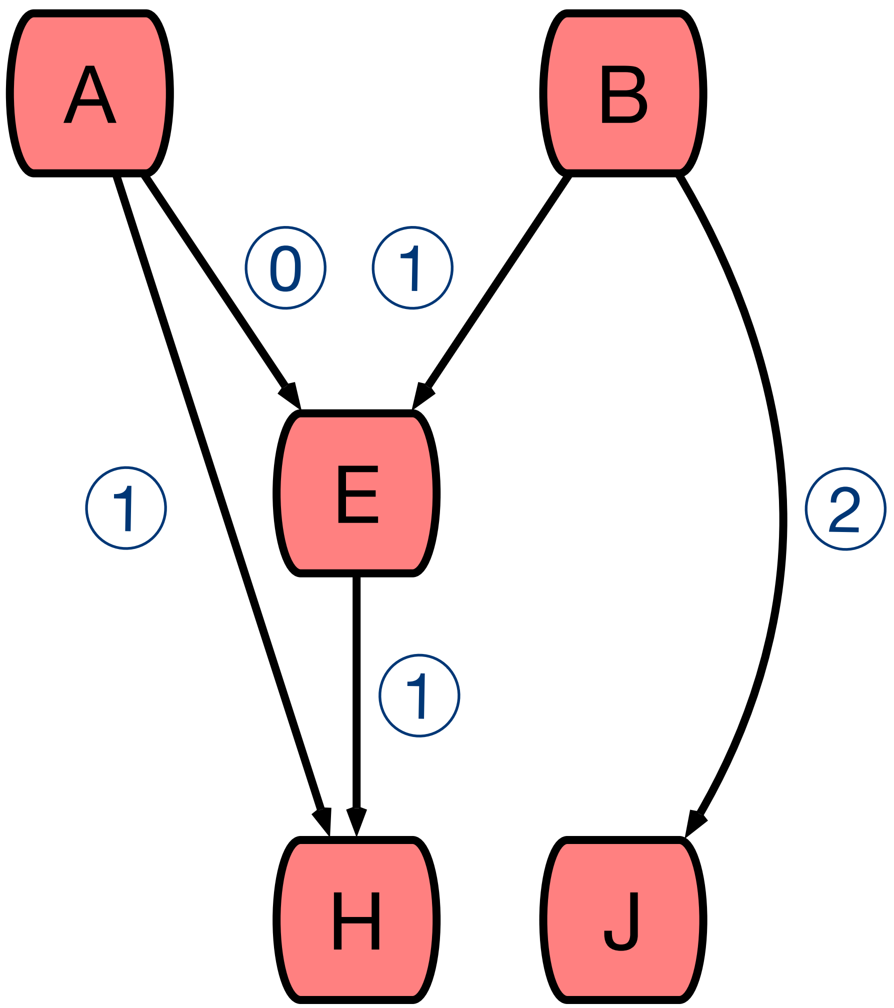
- Construct edges between reachable critical operations
  - need to subtract source operation's delay

- Backedges are copied over from original instance



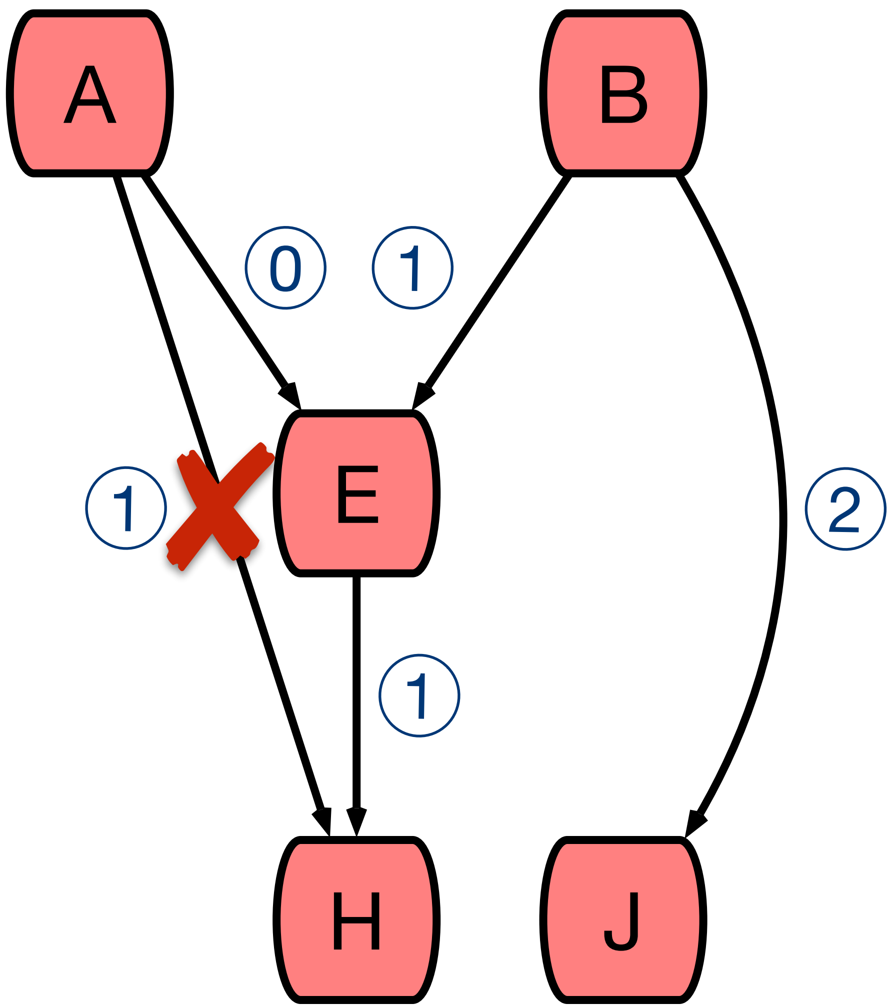
1c) Filter Redundant Edges

- Precedence constraints may be modelled transitively



1c) Filter Redundant Edges

- Precedence constraints may be modelled transitively
- Find and remove such edges



2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler

2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler
  - Here: using **exact** formulations with integer linear programs (ILP)

2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler
  - Here: using **exact** formulations with integer linear programs (ILP)
  - Eichenberger & Davidson (1997) ***„ED“***

2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler
  - Here: using **exact** formulations with integer linear programs (ILP)
  - Eichenberger & Davidson (1997) ***„ED"***
  - Oppermann et al. (2016) ***„Moovac"***

2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler
  - Here: using **exact** formulations with integer linear programs (ILP)
  - Eichenberger & Davidson (1997) ***„ED“***
  - Oppermann et al. (2016) ***„Moovac“***

→ different approaches to model operations’ start times and resource limits

3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations

3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations

  → feasible/optimal for **original** instance

3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations

  → feasible/optimal for **original** instance

- To be computed: start times for non-critical operations

3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations

  → feasible/optimal for **original** instance

- To be computed: start times for non-critical operations
  - fix start times of critical operations in **original** instance, and schedule!

3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations

  → feasible/optimal for **original** instance

- To be computed: start times for non-critical operations
  - fix start times of critical operations in **original** instance, and schedule!

  → easy (polynomially) to solve, because no longer resource-constrained

Agenda

- ☑ (Short) introduction to modulo scheduling
- ☑ Proposed preprocessing approach
- ▶ **Results and insights**

Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac

Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
  - 167 other loops were scheduled optimally with both approaches in less time

Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
  - 167 other loops were scheduled optimally with both approaches in less time

- **Time limit** (per candidate II): 3 minutes

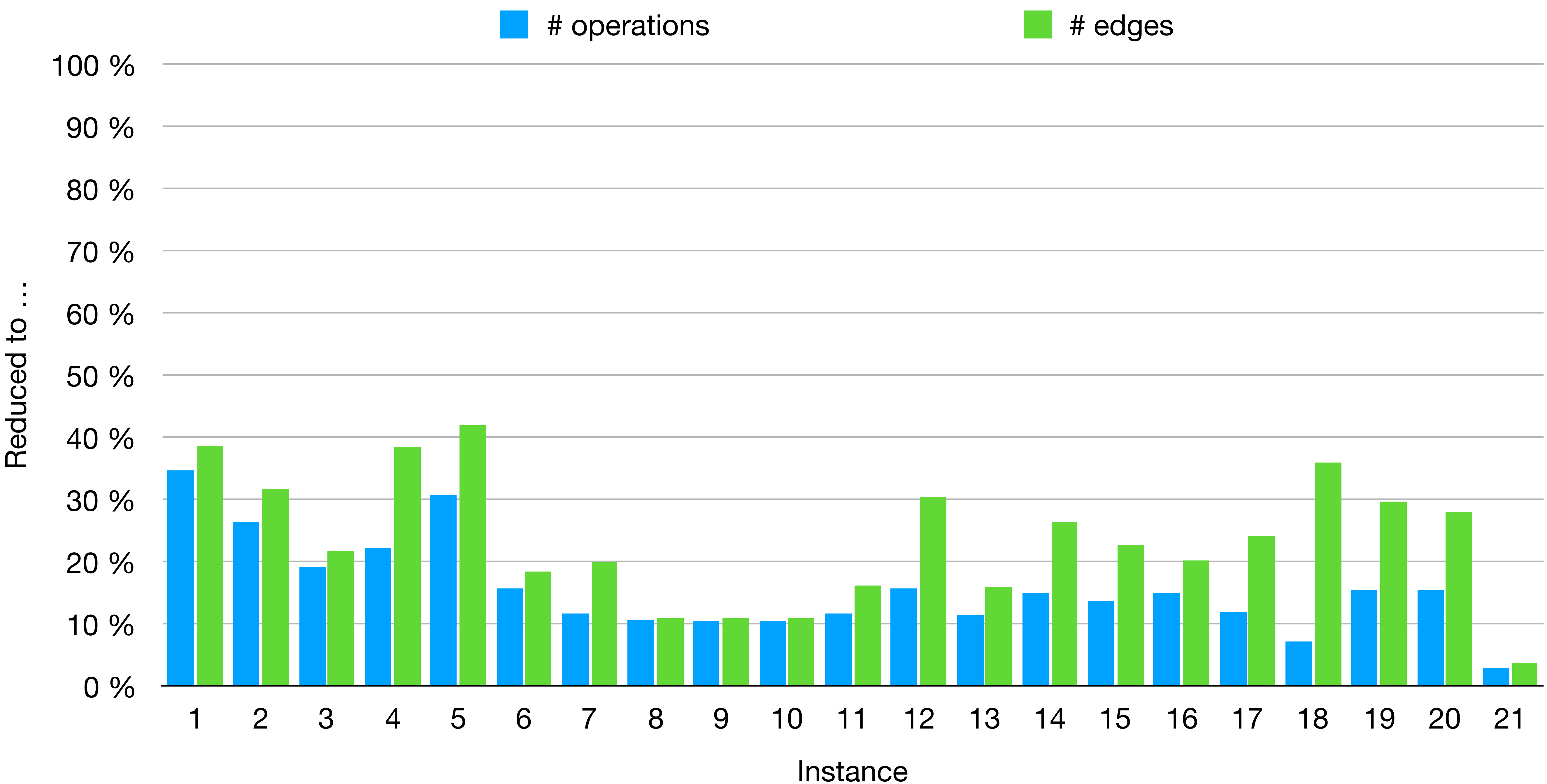
Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
  - 167 other loops were scheduled optimally with both approaches in less time
- **Time limit** (per candidate II): 3 minutes
- **CPLEX** 12.6.3 as ILP solver, 8x multithreaded

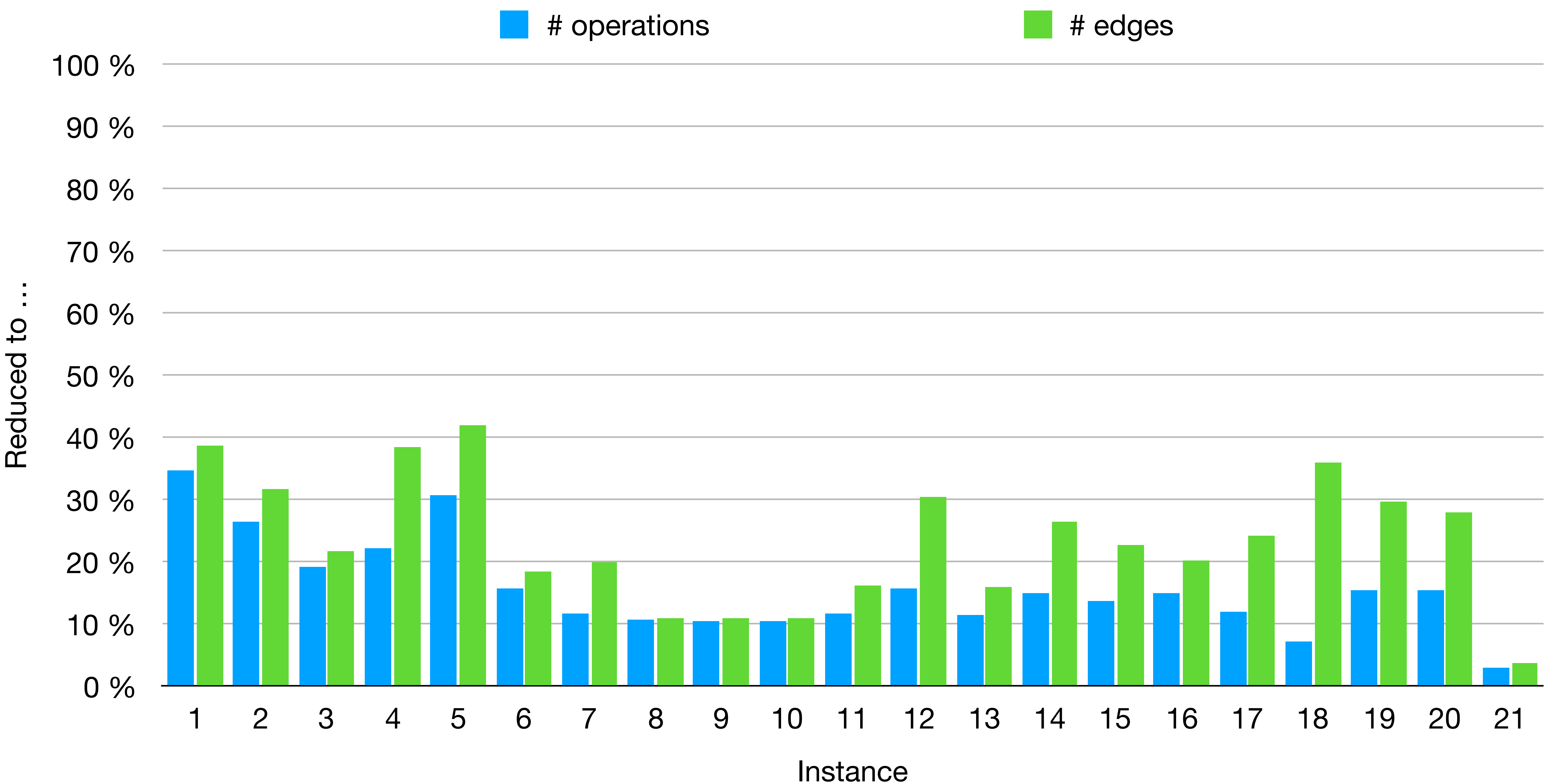
Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
  - 167 other loops were scheduled optimally with both approaches in less time
- **Time limit** (per candidate II): 3 minutes
- **CPLEX** 12.6.3 as ILP solver, 8x multithreaded
- Ran on 24-core Xeon E5-2680 v3, 2.8 GHz, 64 GB RAM

Results: Graph Reduction

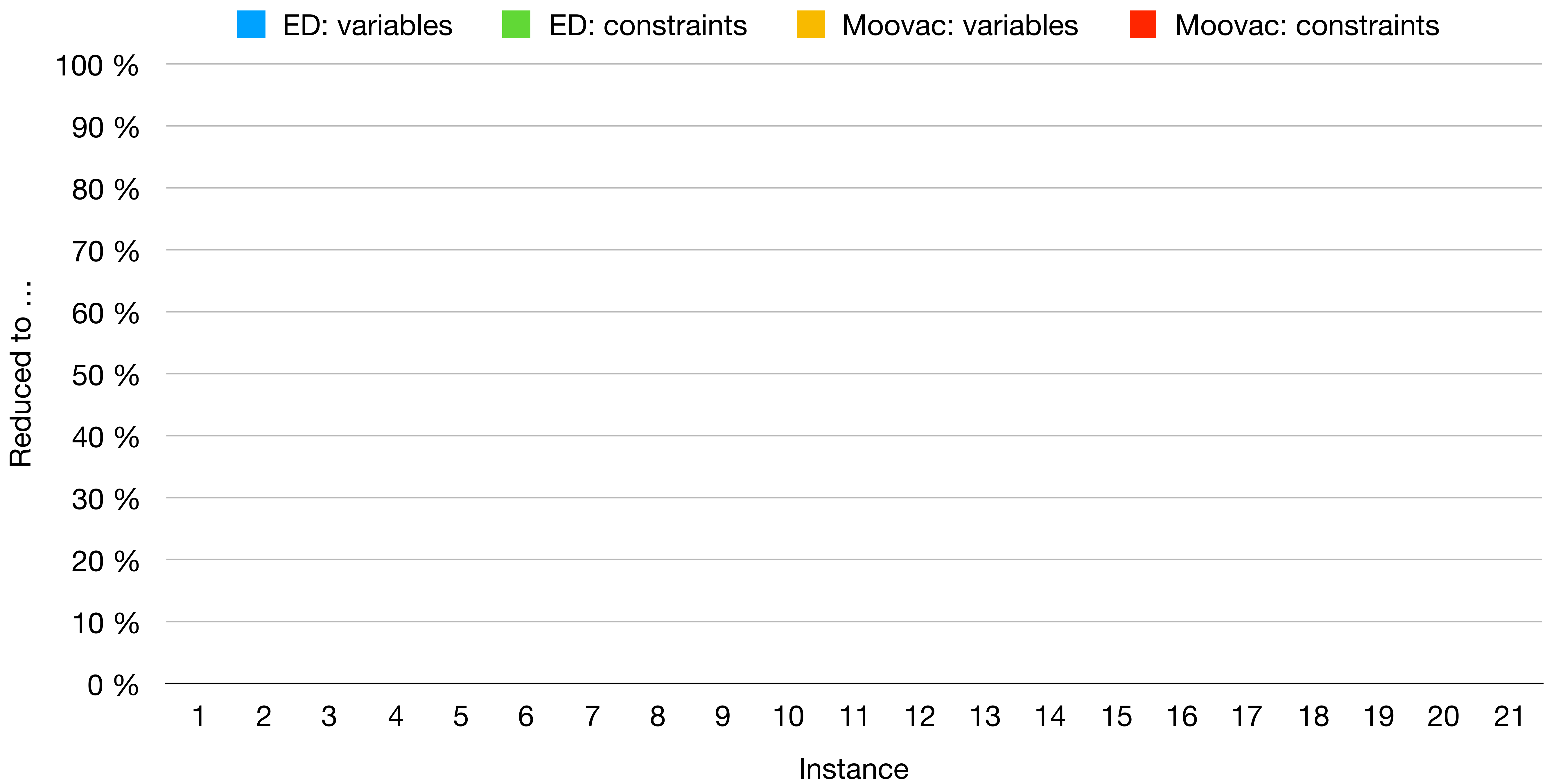


Results: Graph Reduction



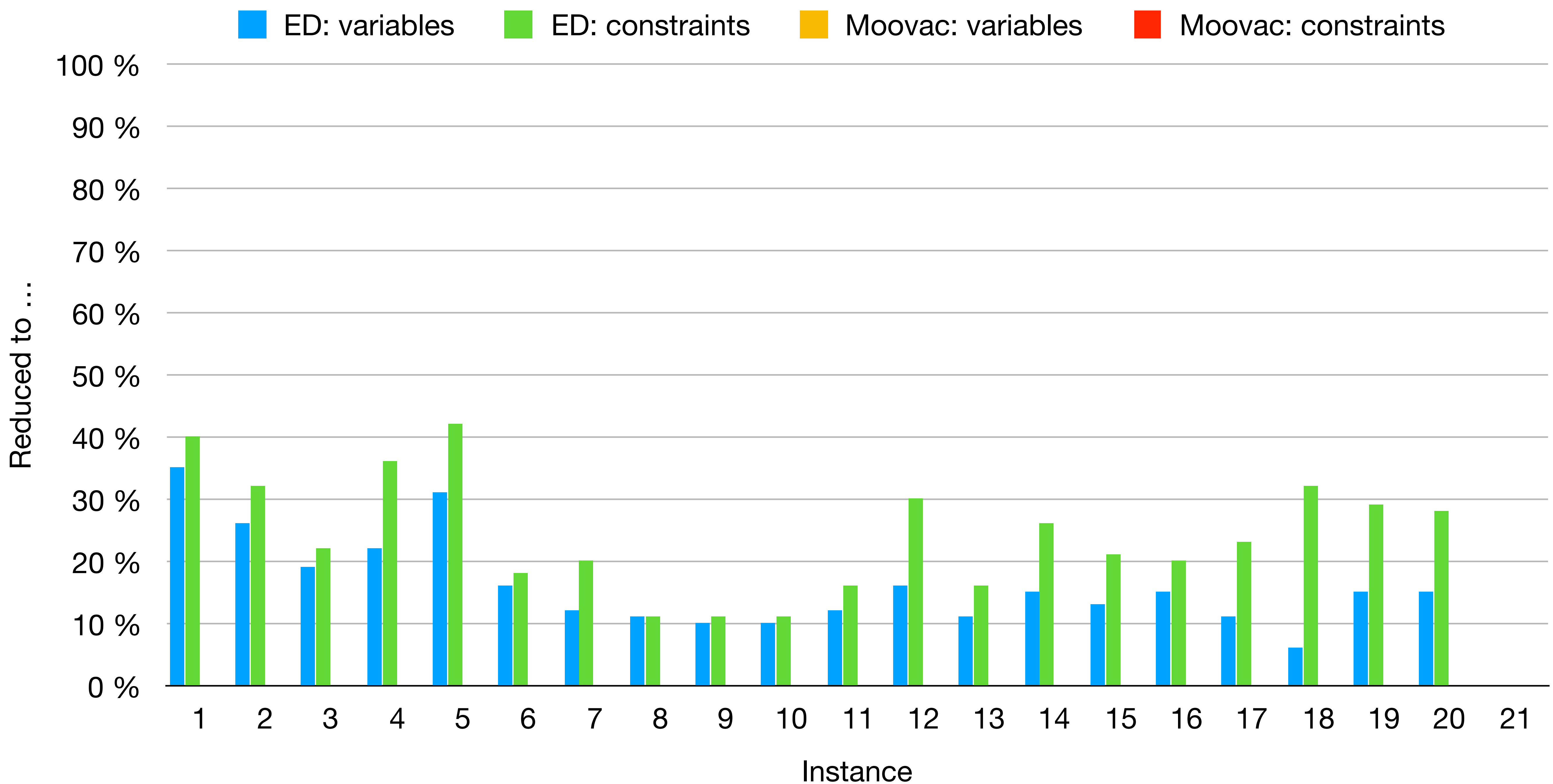
- Time for complexity reduction: **always < 0.5 sec**

Results: ILP Reduction



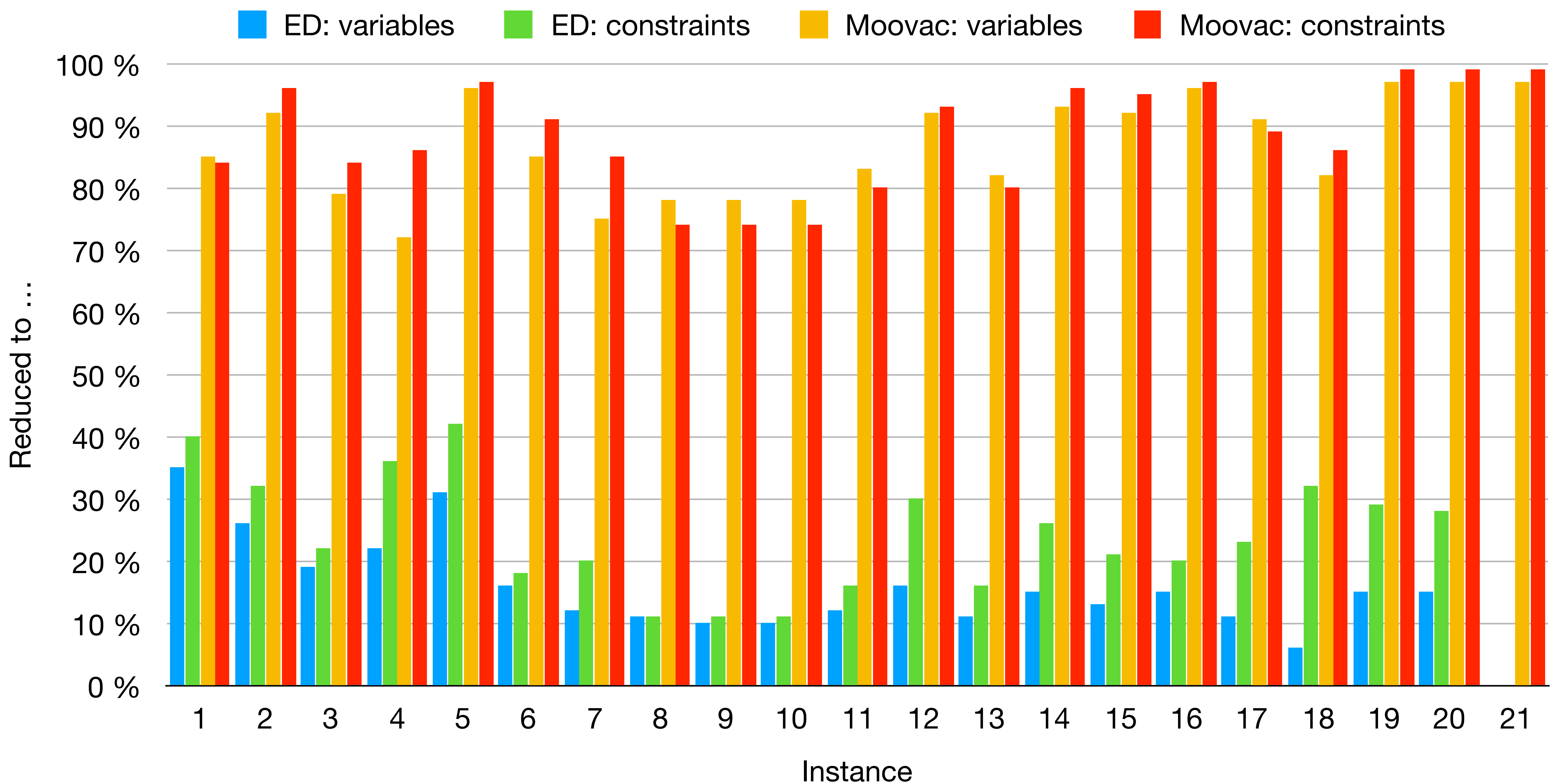
Mean #
Variables
Constraints

Results: ILP Reduction



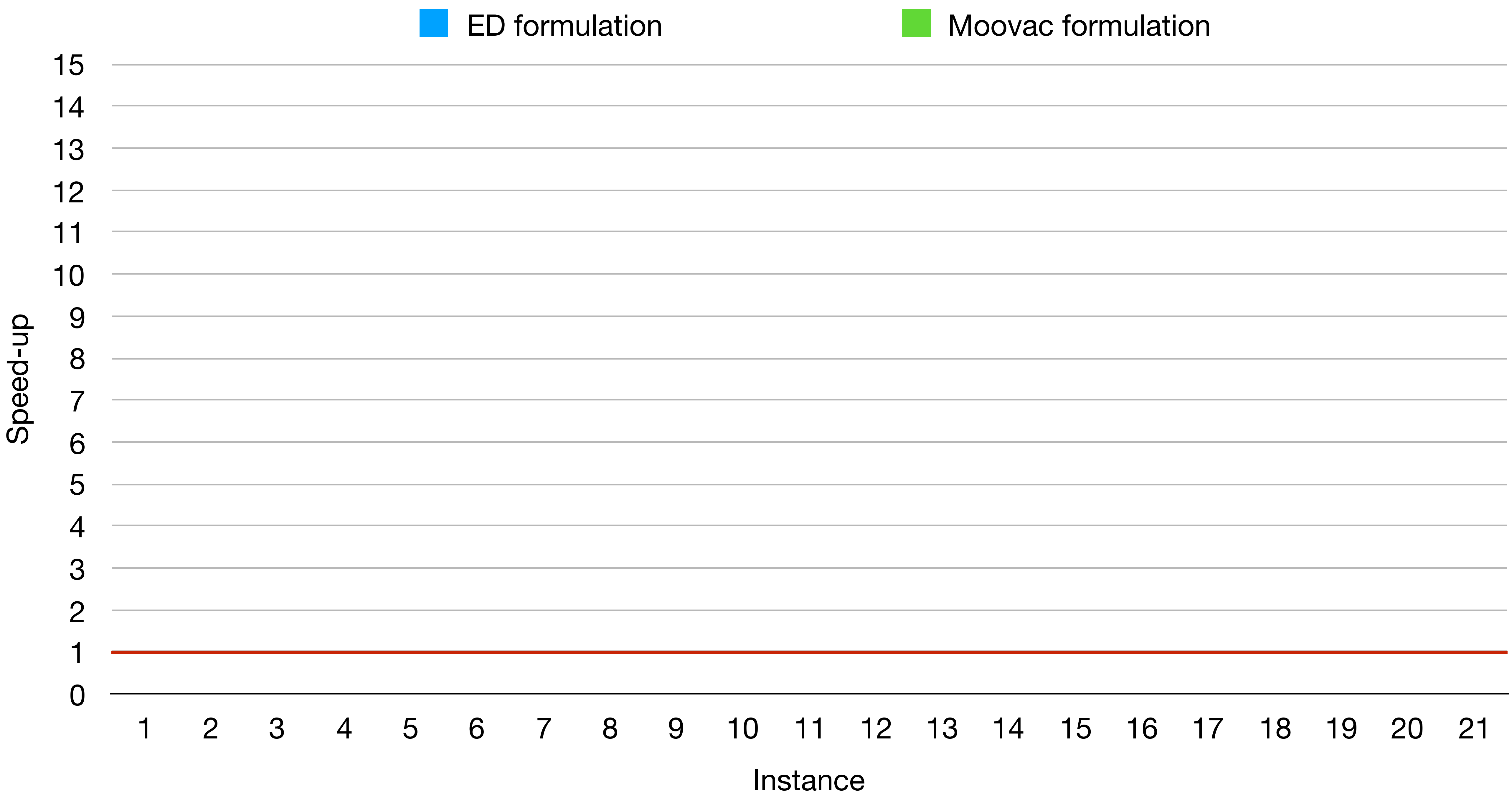
Mean #	ED	ED (red.)
Variables	37.2 k	5.1 k
Constraints	113.4 k	26.7 k

Results: ILP Reduction



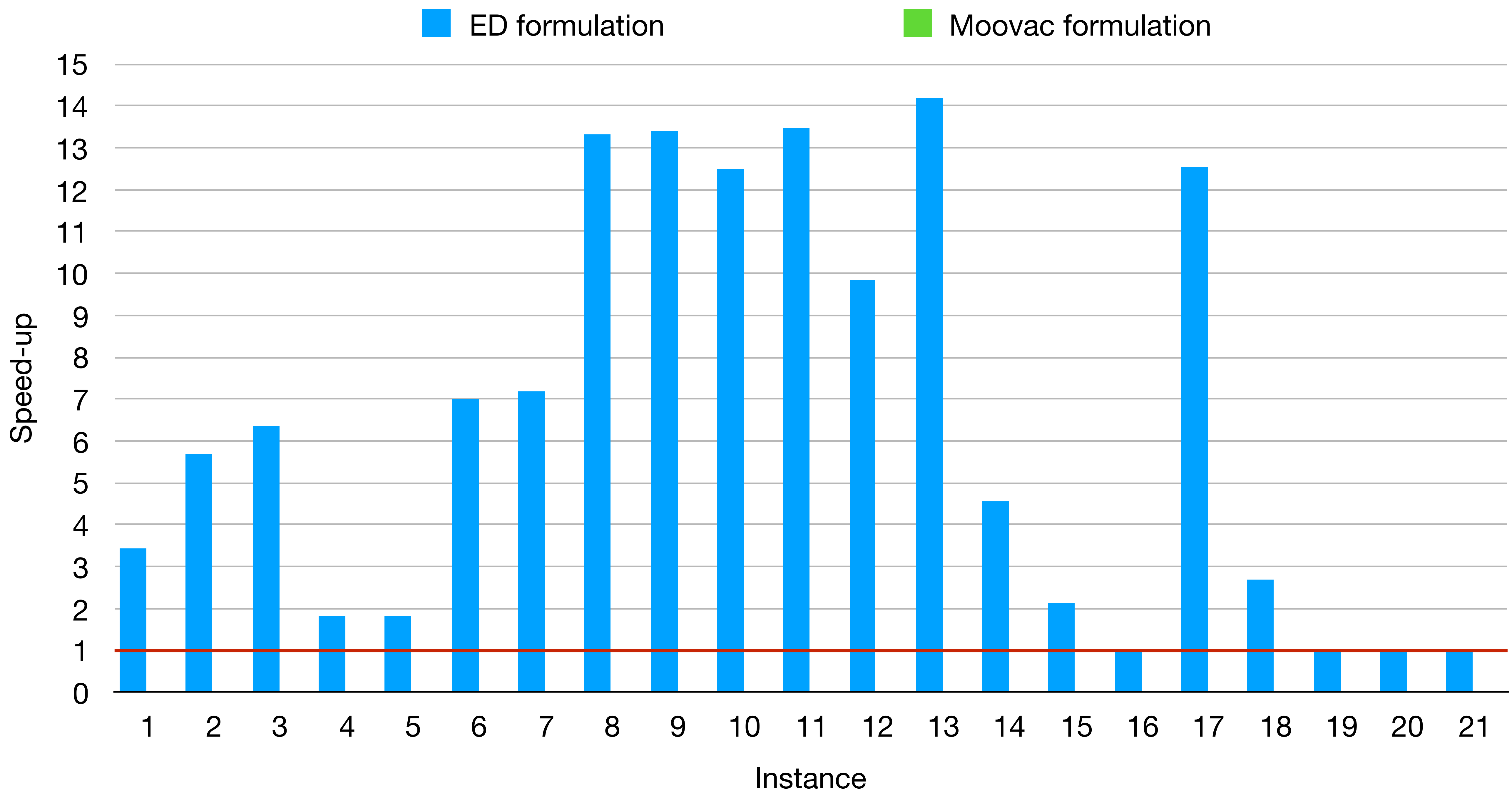
Mean #	ED	ED (red.)	Moovac	Moovac (red.)
Variables	37.2 k	5.1 k	8.4 k	7.9 k
Constraints	113.4 k	26.7 k	28.4 k	27.1 k

Results: Runtime



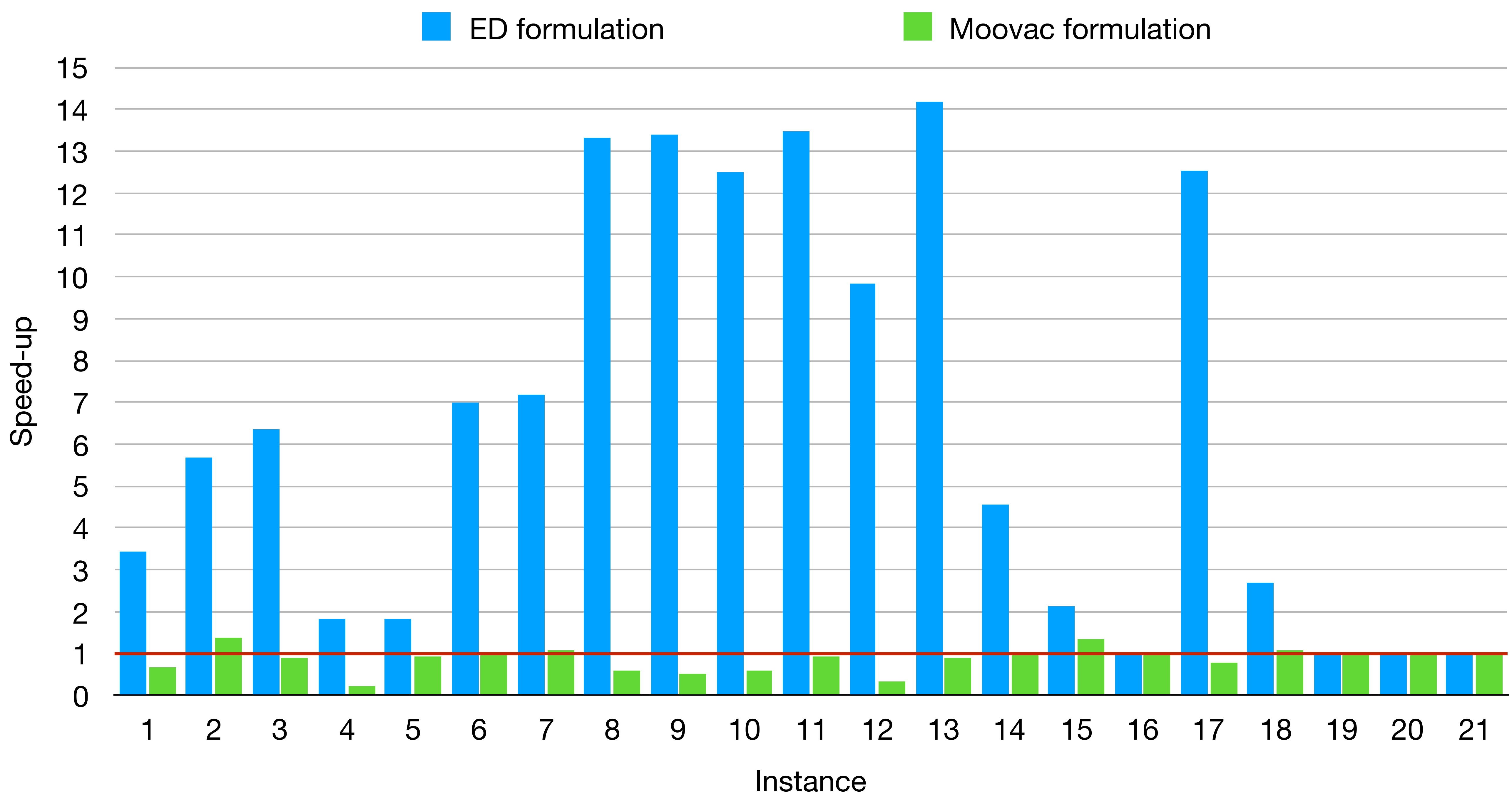
Accumulated runtime
Speed-up (geomean)

Results: Runtime



	ED	ED (red.)
Accumulated runtime	328 min	268 min
Speed-up (geomean)		4.37x

Results: Runtime



	ED	ED (red.)	Moovac	Moovac (red.)
Accumulated runtime	328 min	268 min	290 min	285 min
Speed-up (geomean)		4.37x		0.8x

Results: Solution Quality

# instances	ED	ED (red.)	Moovac	Moovac (red.)
optimal	11	14	13	12
optimal II	2	-	2	3
feasible	3	4	3	3
no solution	5	3	3	3

Results: Solution Quality

# instances	ED	ED (red.)	Moovac	Moovac (red.)
optimal	11	14	13	12
optimal II	2	-	2	3
feasible	3	4	3	3
no solution	5	3	3	3

- More instances are tractable for ED formulation

Results: Solution Quality

# instances	ED	ED (red.)	Moovac	Moovac (red.)
optimal	11	14	13	12
optimal II	2	-	2	3
feasible	3	4	3	3
no solution	5	3	3	3

- More instances are tractable for ED formulation

Results: Solution Quality

# instances	ED	ED (red.)	Moovac	Moovac (red.)
optimal	11	14	13	12
optimal II	2	-	2	3
feasible	3	4	3	3
no solution	5	3	3	3

- More instances are tractable for ED formulation
- Again, minor regression for Moovac

Discussion

- **ED:** Significant benefits
  - both in terms of **runtime** and **solution quality**
  - additional effort for problem reduction is negligible

Discussion

- **ED:** Significant benefits
  - both in terms of **runtime** and **solution quality**
  - additional effort for problem reduction is negligible

- **Moovac:** ILP complexity dominated by resource-limited operations
  - not enough reduction potential to offset ILP solvers' *„performance variability"*

Discussion

- **ED:** Significant benefits
  - both in terms of **runtime** and **solution quality**
  - additional effort for problem reduction is negligible

- **Moovac:** ILP complexity dominated by resource-limited operations
  - not enough reduction potential to offset ILP solvers' *„performance variability"*

- Both now **much closer** performance-wise
  - seem to **complement** each other

Conclusion & Outlook

- *Can't we just simplify the problem?*

Conclusion & Outlook

- *Can't we just simplify the problem?*
  - Yes!

Conclusion & Outlook

- *Can't we just simplify the problem?*
  - Yes!
  - other, similar ILP formulations exist

Conclusion & Outlook

- *Can’t we just simplify the problem?*
  - Yes!
  - other, similar ILP formulations exist
- Long-term goal: an **oracle**
  - select the „right“ modulo scheduler for a given instance

Conclusion & Outlook

- *Can’t we just simplify the problem?*
  - Yes!
  - other, similar ILP formulations exist

- Long-term goal: an **oracle**
  - select the „right“ modulo scheduler for a given instance

  → important to have different schedulers that scale roughly the same

Thank you!

oppermann@esa.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

KIT Karlsruhe Institute of Technology

THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND