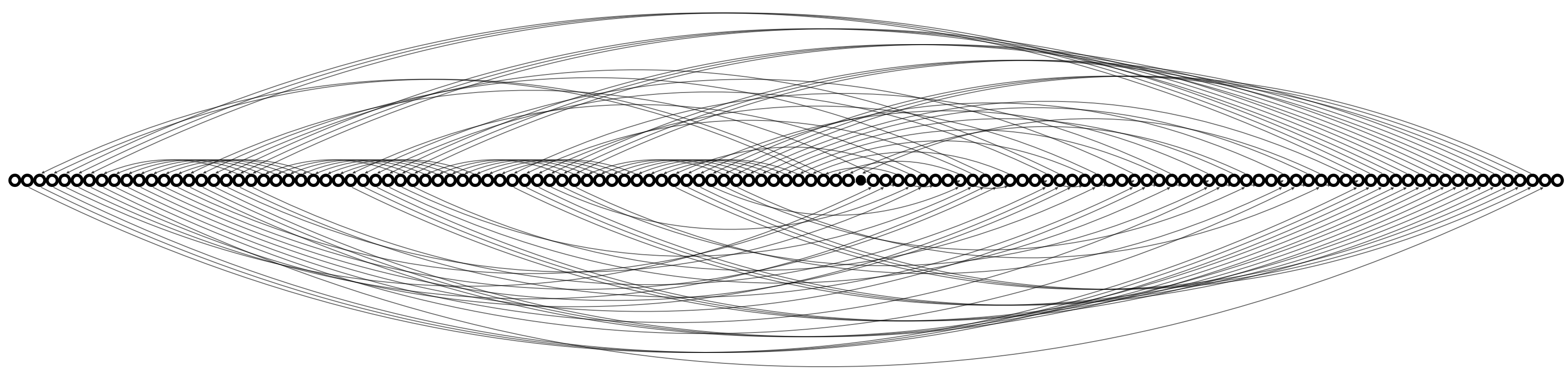
Transiently Secure Network Updates



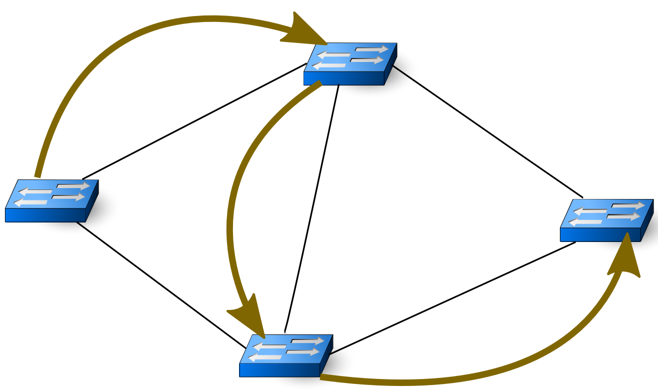
Arne Ludwig[1], Szymon Dudycz[2], Matthias Rost[1], Stefan Schmid[3]

TU Berlin[1], University of Wroclaw[2], Aalborg University[3]

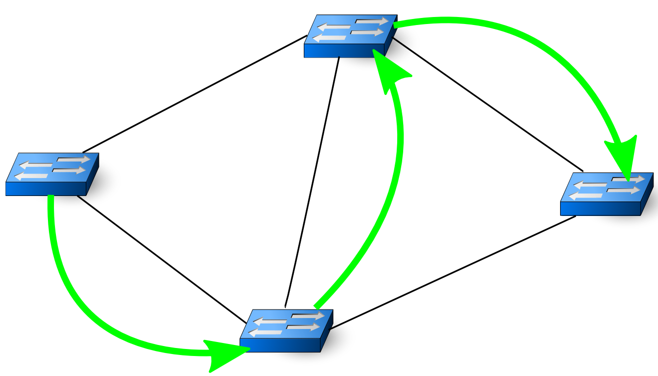
Network Updates

How to transition from old to new path?

old path



new path



While not discarding any packets!

Network Updates Happen

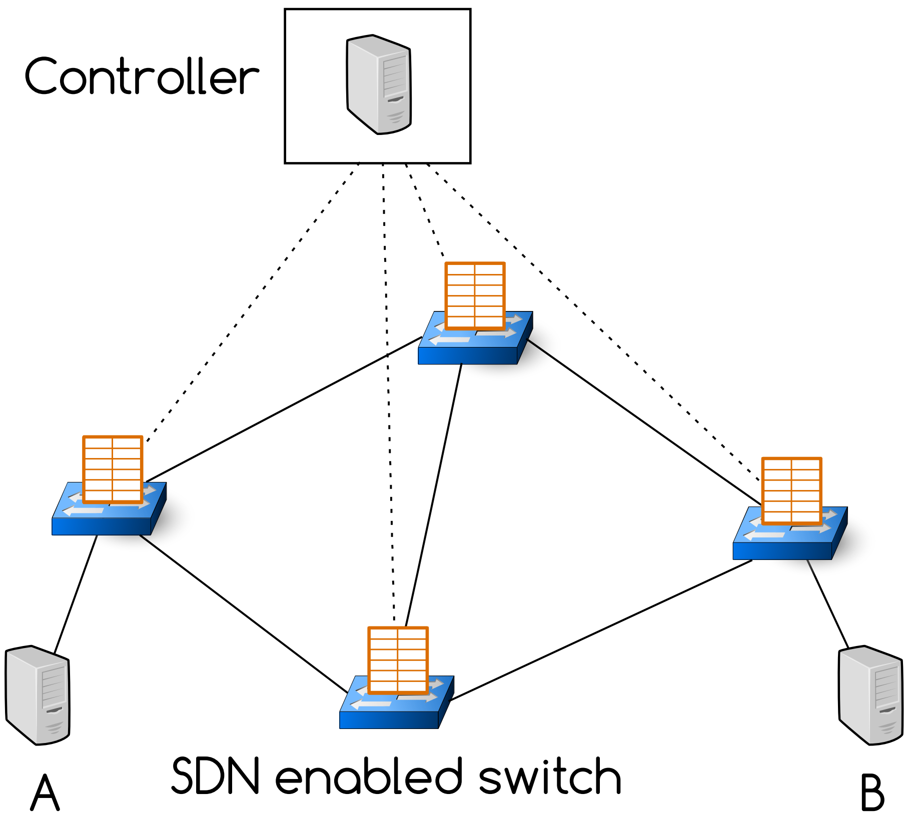
Error prone task

manual updates per device, despite global goals

Misconfiguration on switches that caused a "bridge loop". [2012]

A network change was [...] executed incorrectly [...] re-mirroring storm [2011]

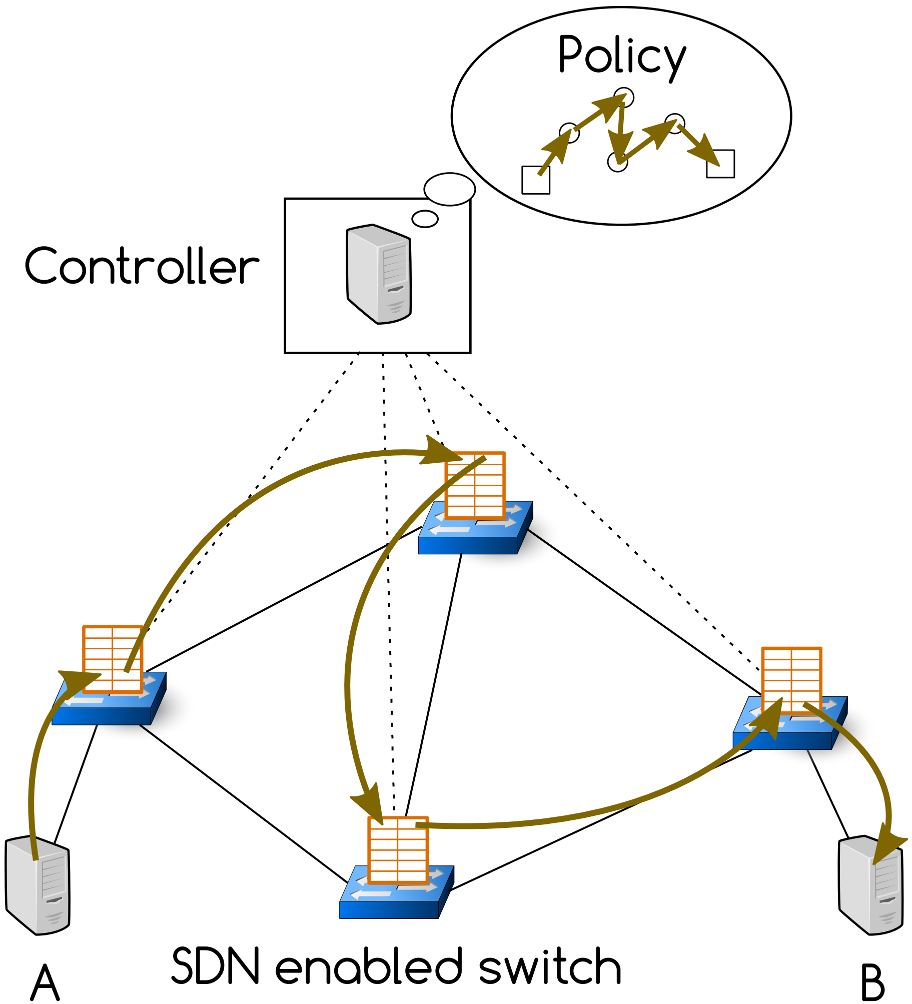
Model



Software-Defined Networking (SDN)

- Separate control from data plane
- Logically centralized network view (controller)

Model



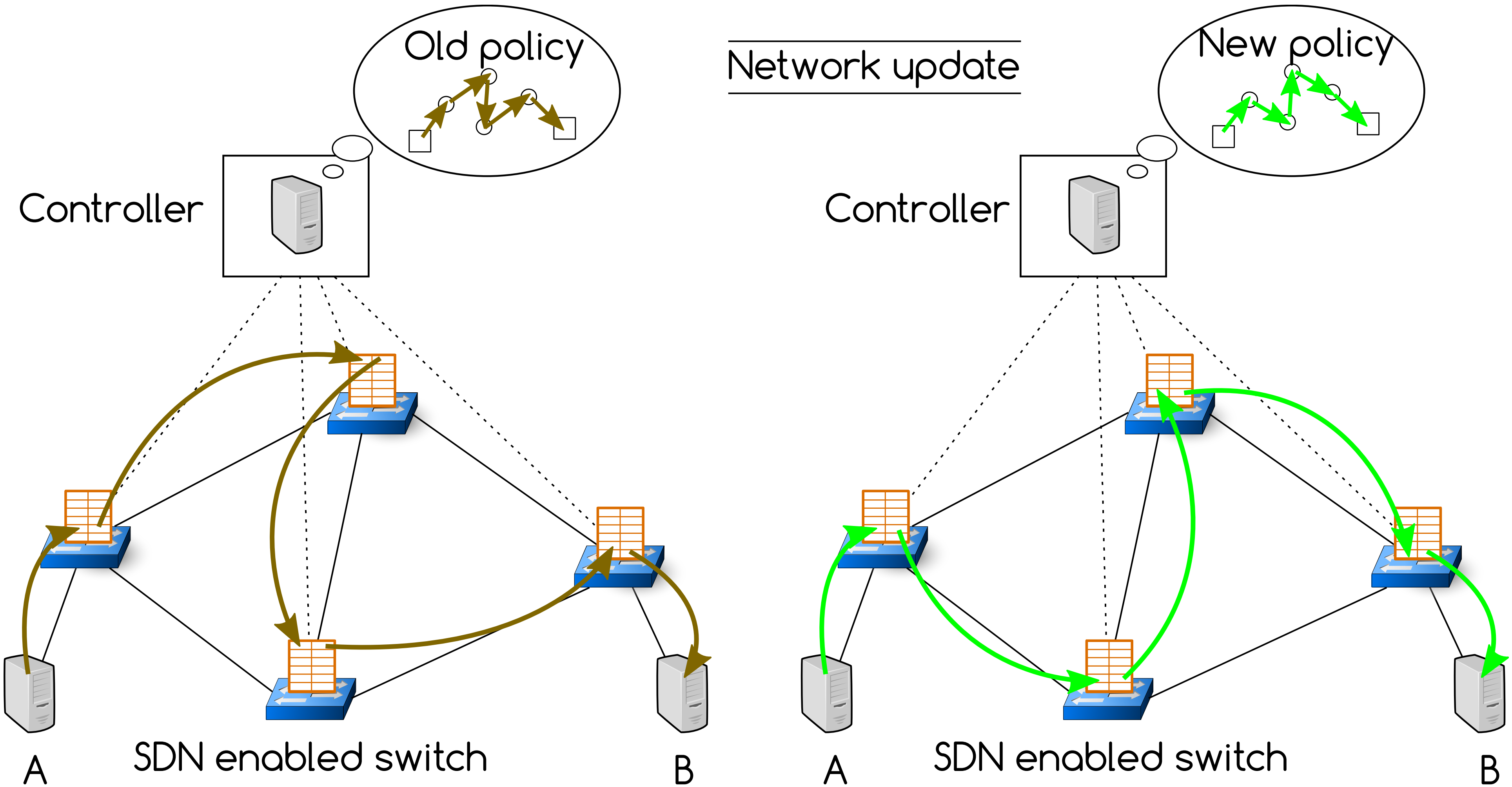
Software-Defined Networking (SDN)

- Separate control from data plane

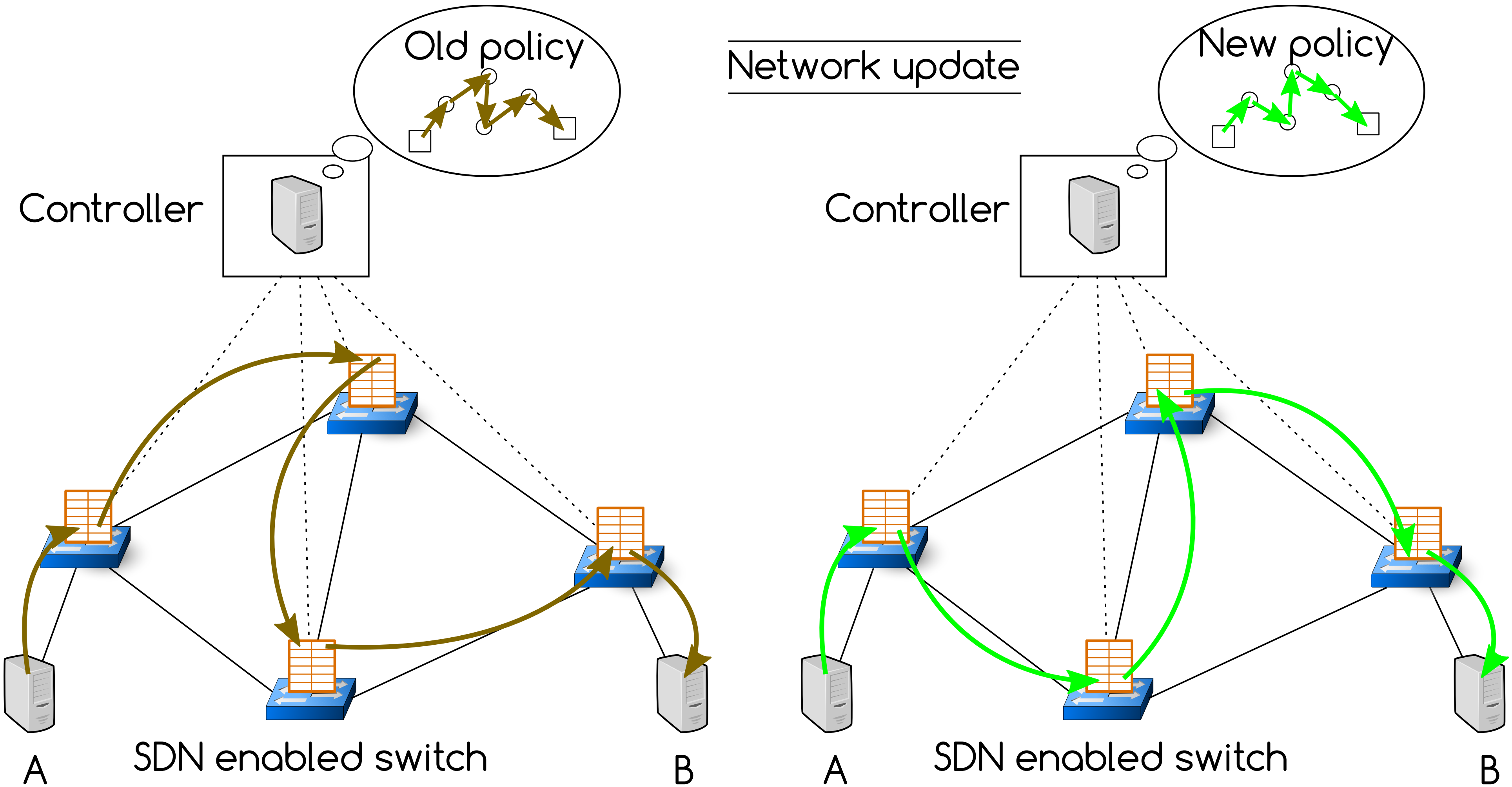
- Logically centralized network view (controller)

- Not only destination based (match-action rules)

Model


Old policy
Controller
SDN enabled switch
A
B
Network update
New policy
Controller
SDN enabled switch
A
B

Model

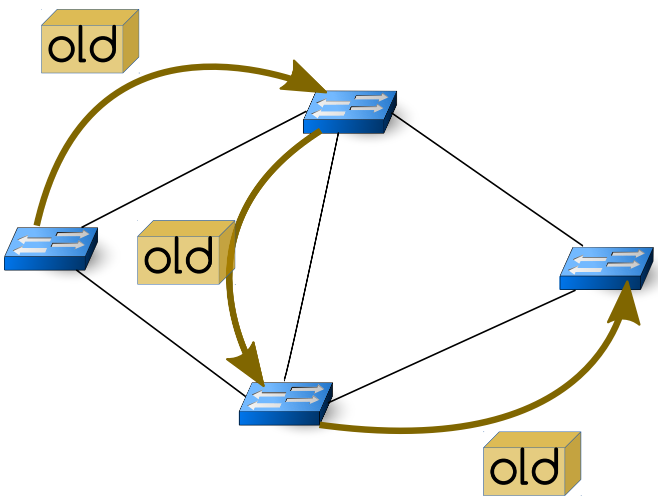

Old policy
Controller
SDN enabled switch
A
B
Network update
New policy
Controller
SDN enabled switch
A
B

Strong Consistency

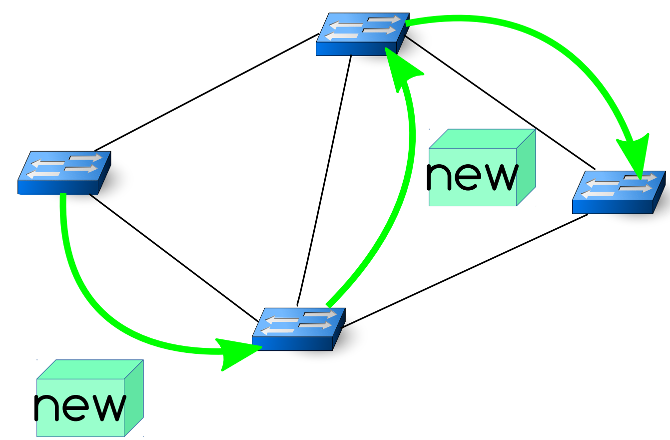
Two-phase commit [REI12] → Either old or new policy

Strong Consistency

Two-phase commit [REI12] → Either old or new policy

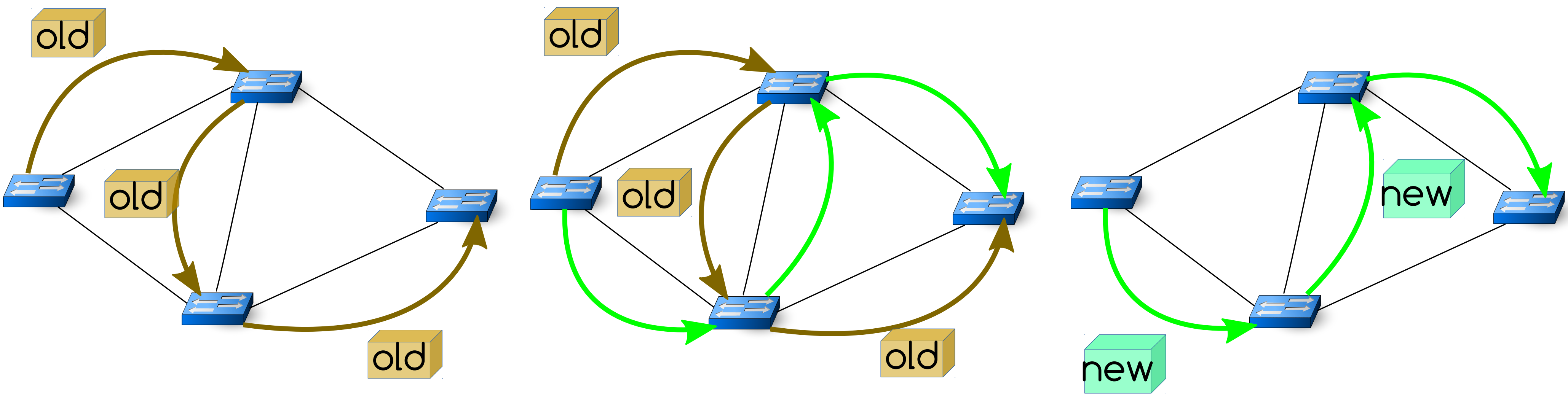


Tagging packets
at ingress port



Strong Consistency

Two-phase commit [REI12] → Either old or new policy



Cons:

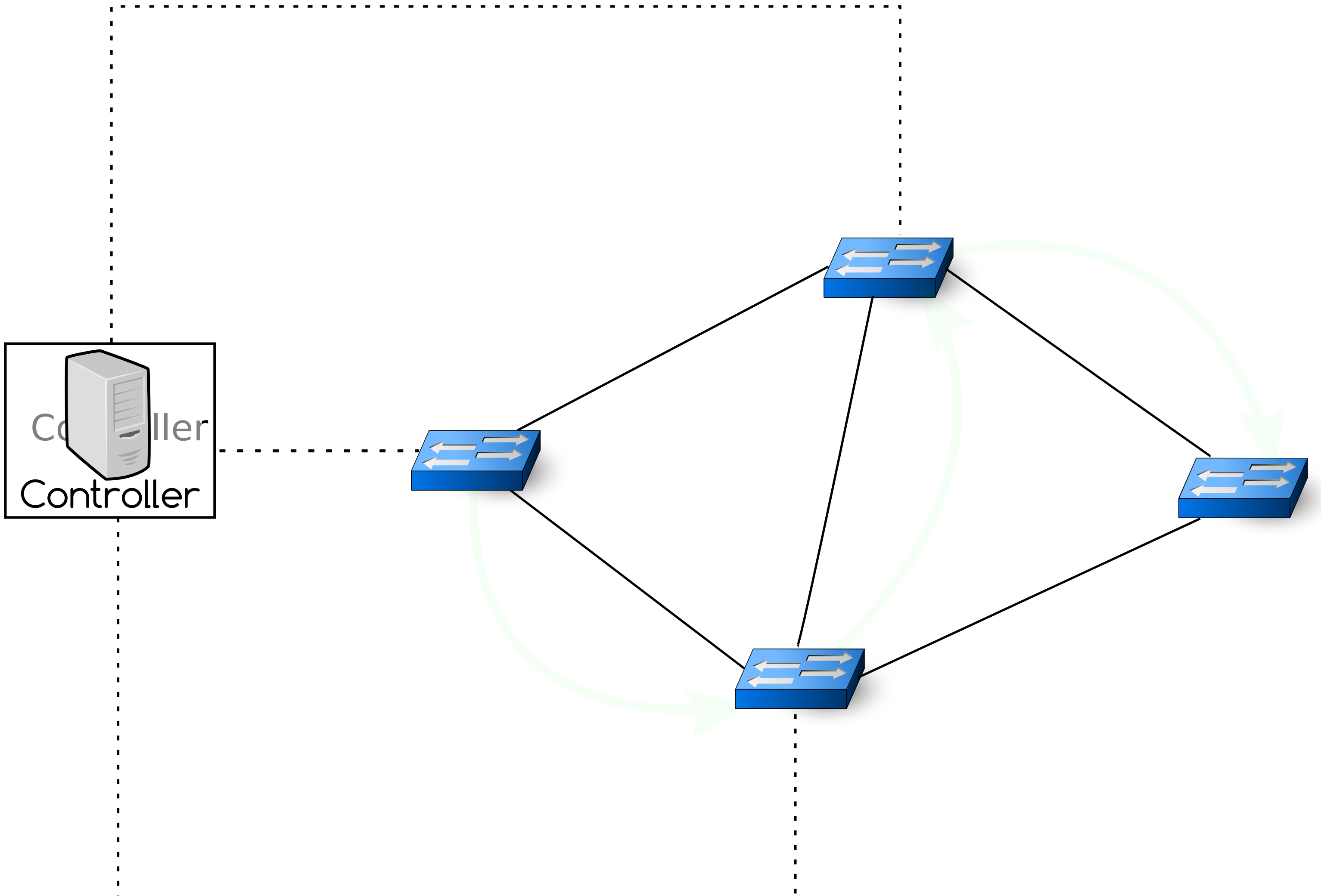
- Needs more switch memory
- **Problematic with middleboxes (changed headers)**

The Challenge: Transiently Secure Updates

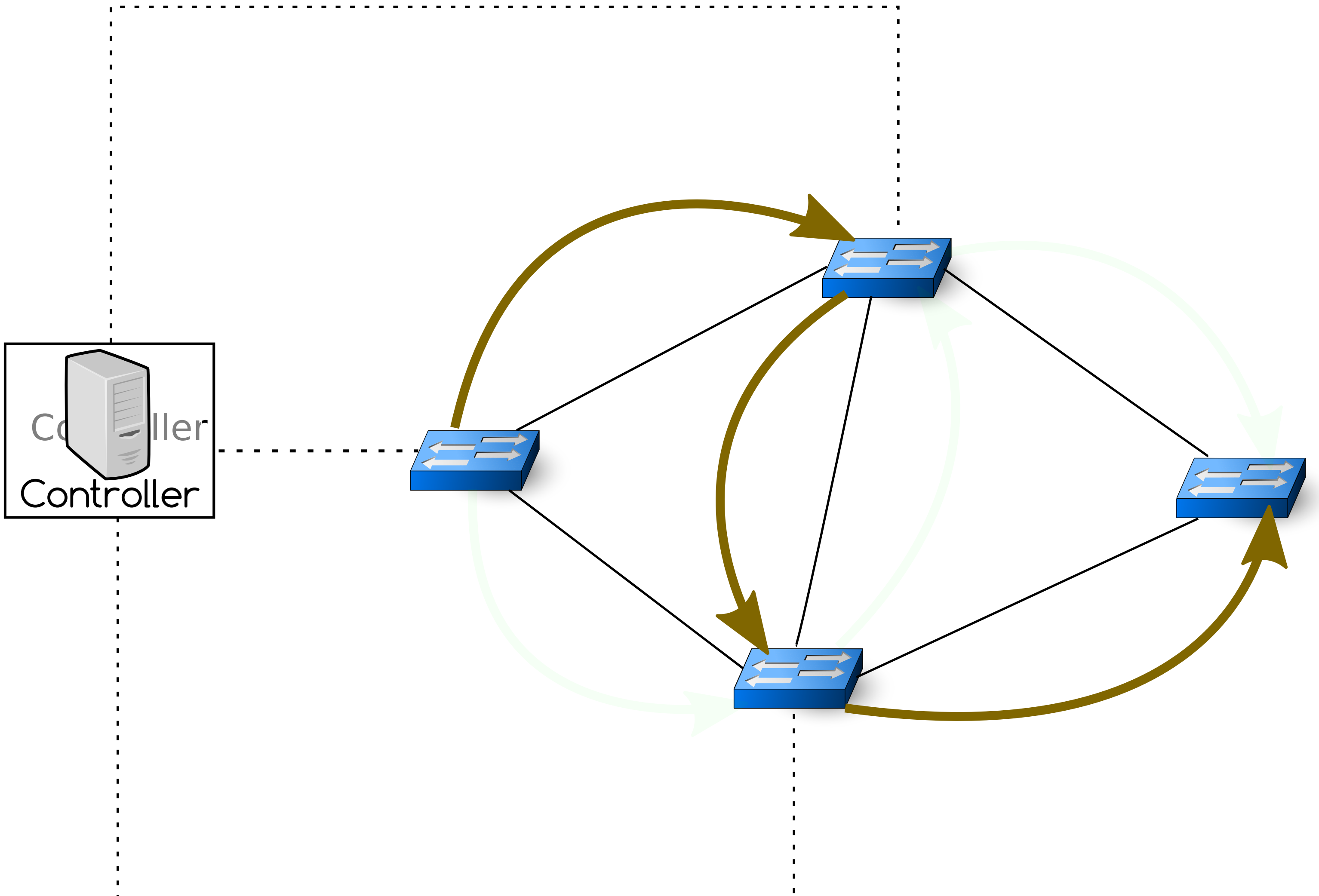
- Consider dynamic updates without tagging [Mahajan et al., HotNets '13]

- Consistent forwarding state needs to be secured:
  - Ensure reachability by forbidding loops
  - **Ensure traversal of waypoints, e.g. firewalls**

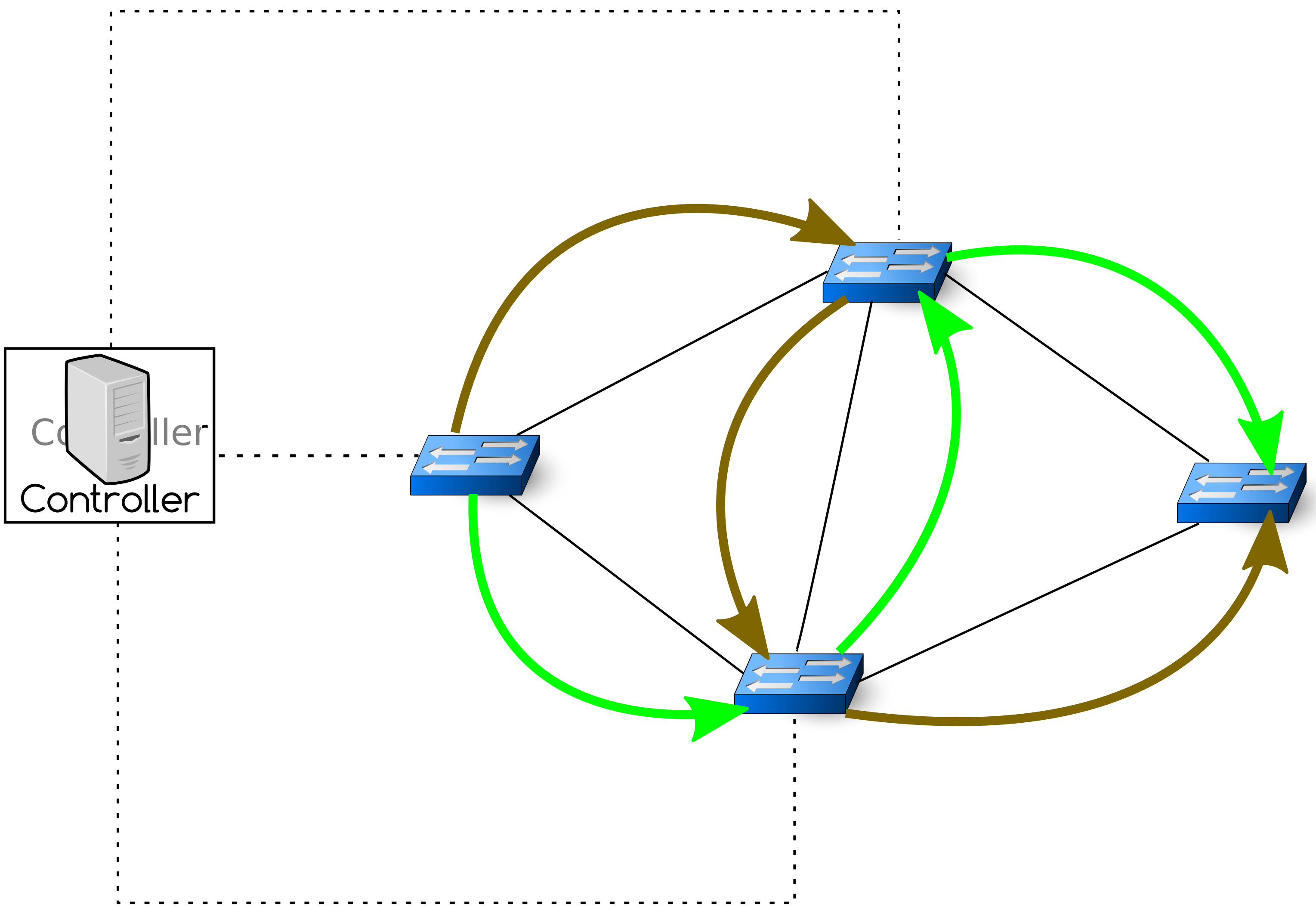
Asynchronous Updates: Timing matters



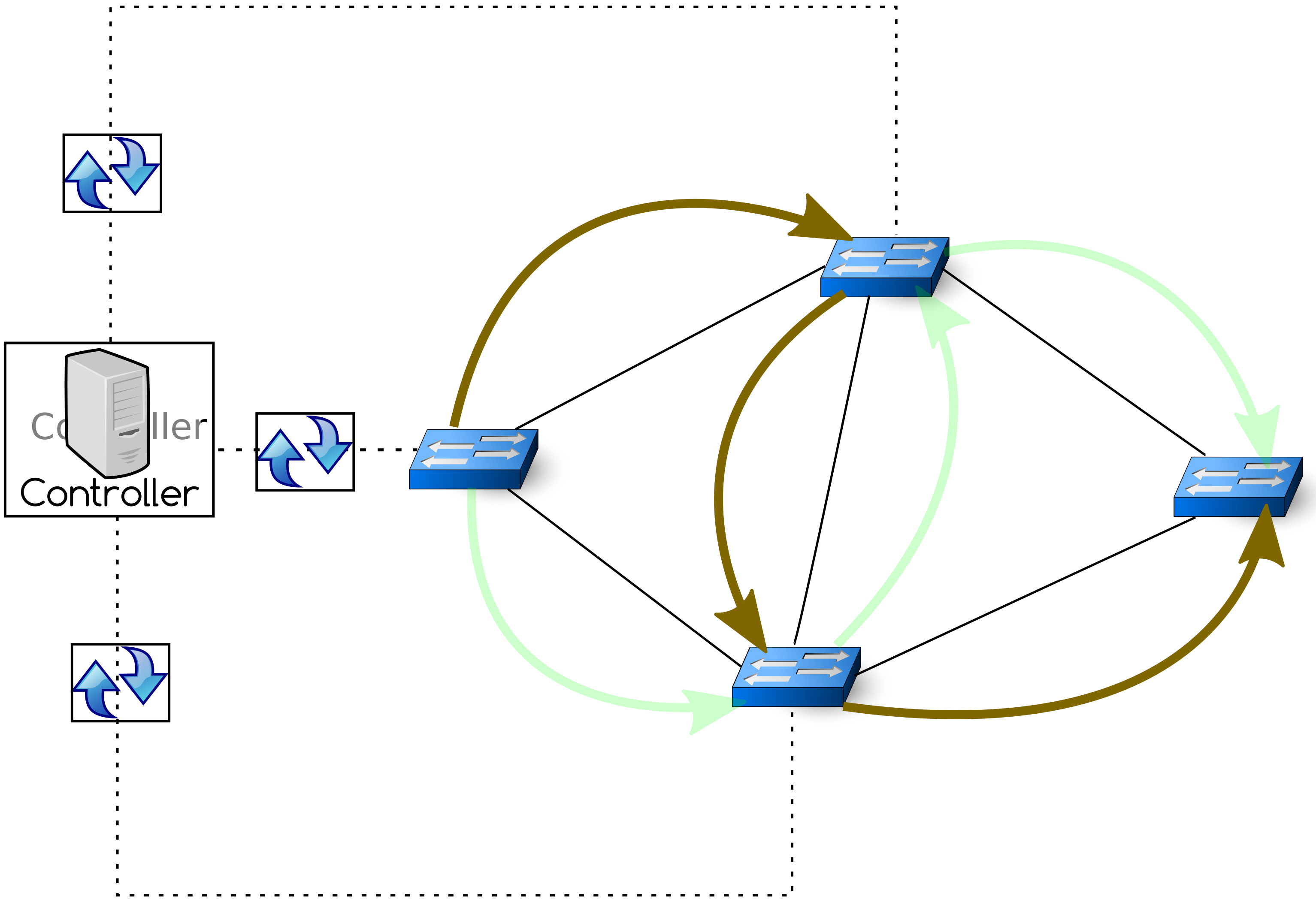
Asynchronous Updates: Timing matters



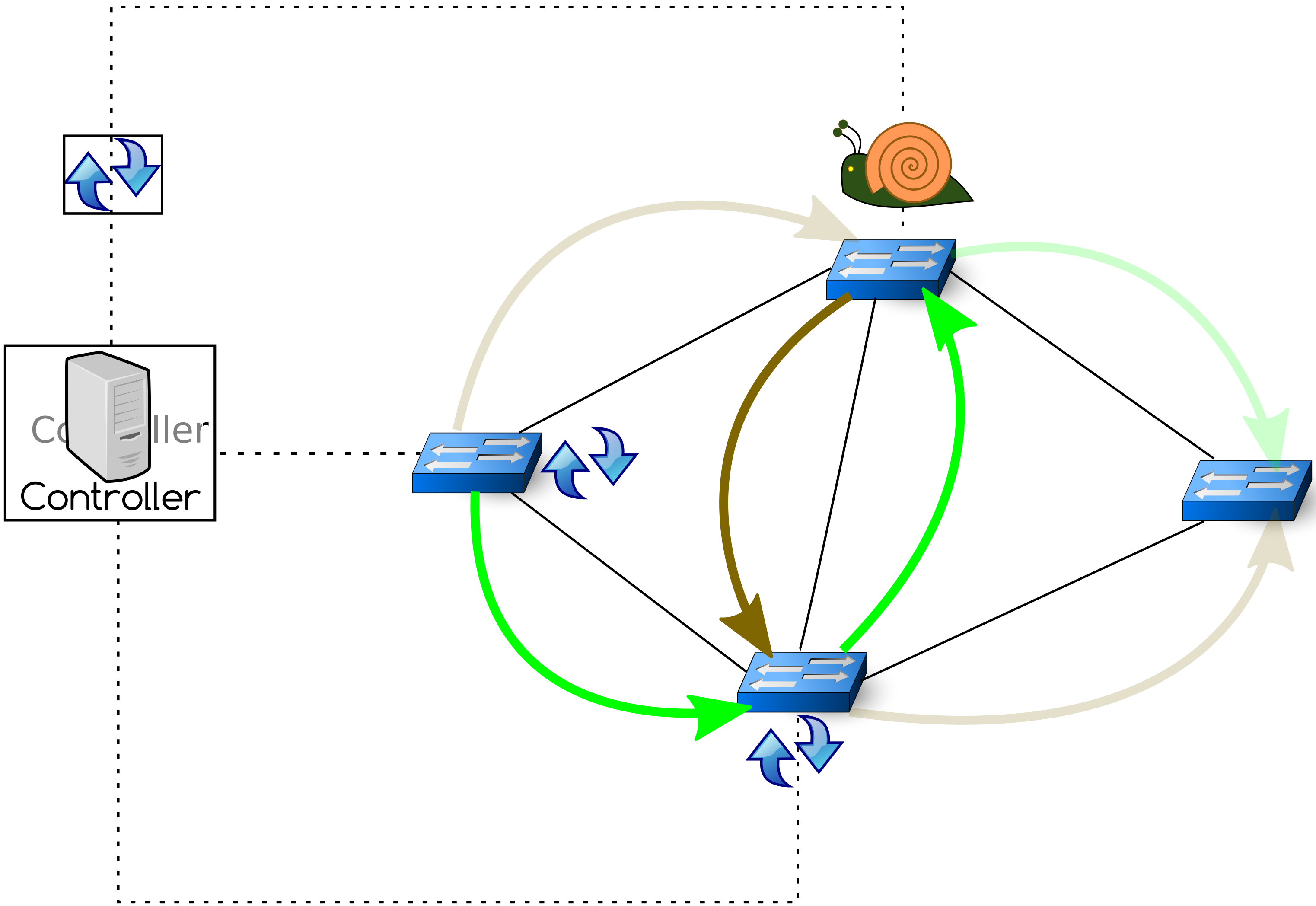
Asynchronous Updates: Timing matters



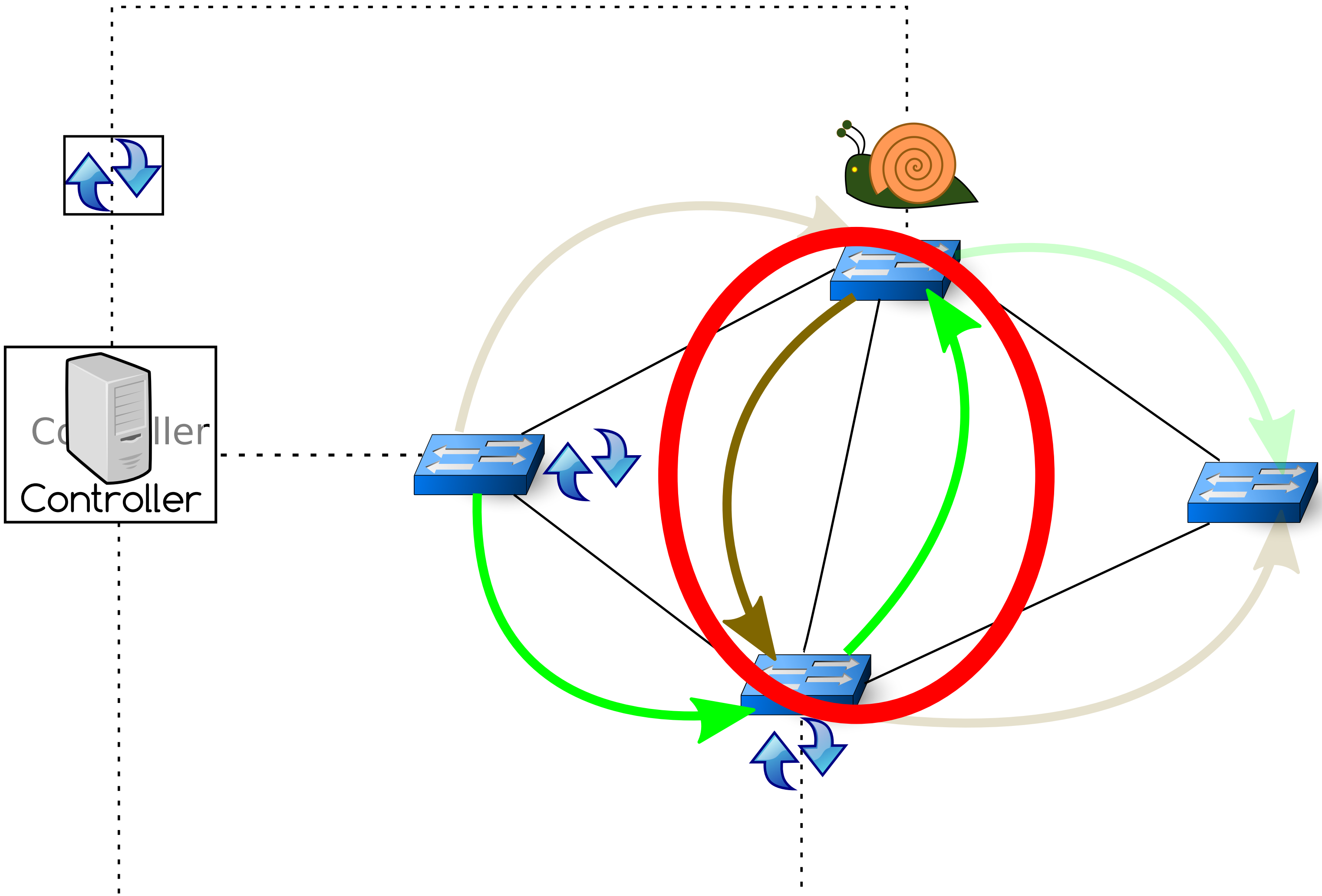
Asynchronous Updates: Timing matters



Asynchronous Updates: Timing matters



Asynchronous Updates: Timing matters

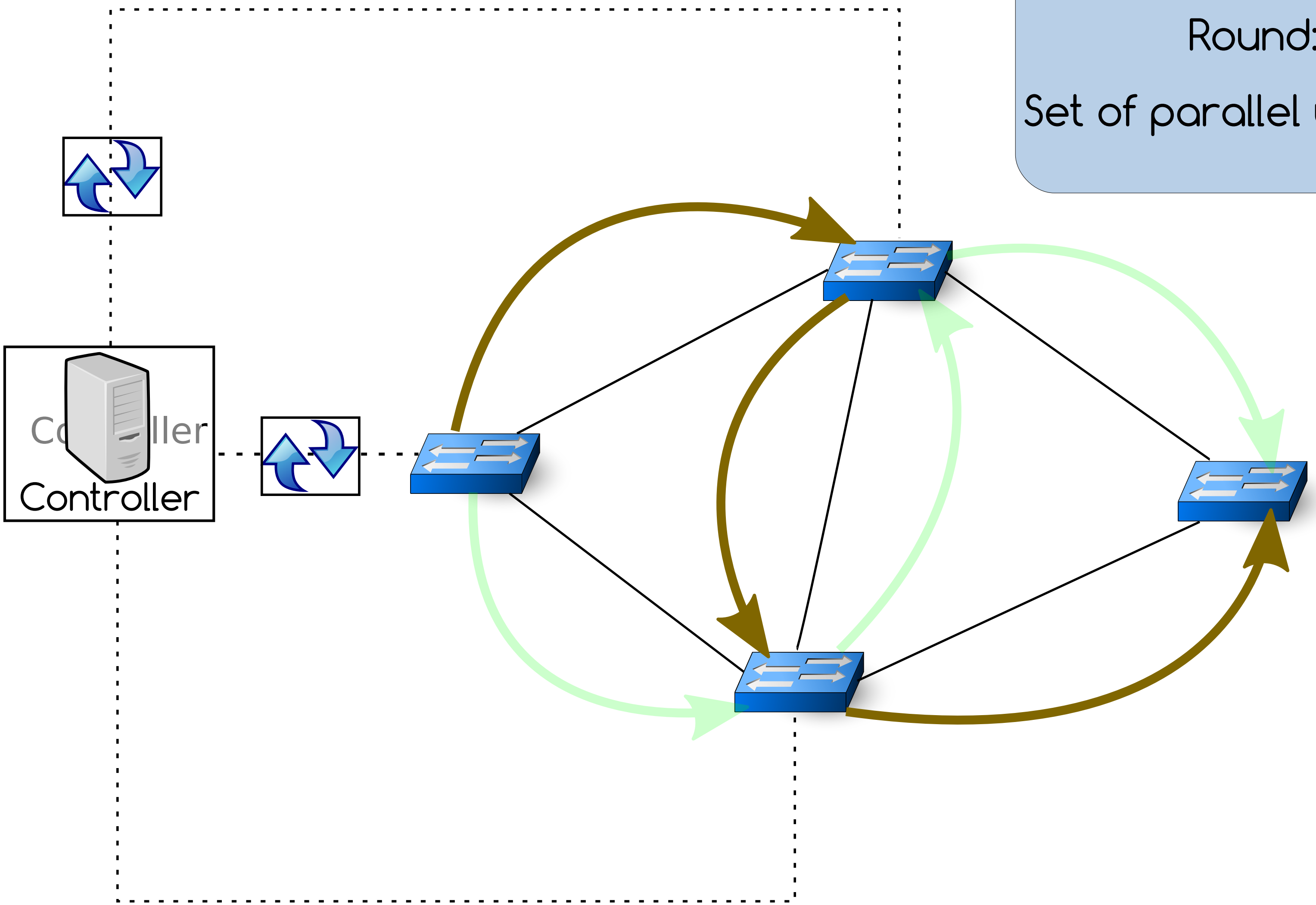


We have to be selective which switches to update

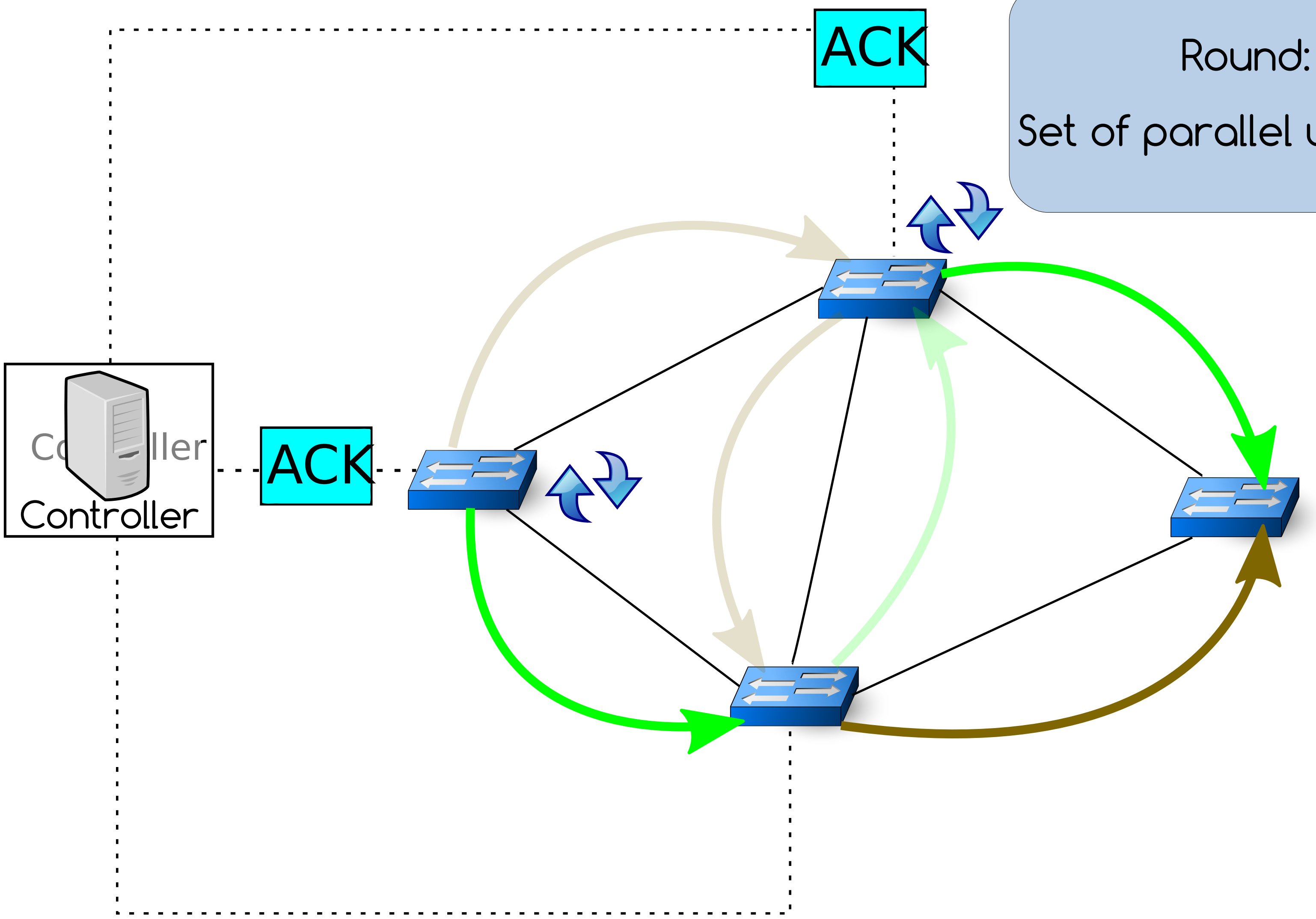
Asynchronous Updates: Round model

Round:

Set of parallel updates.



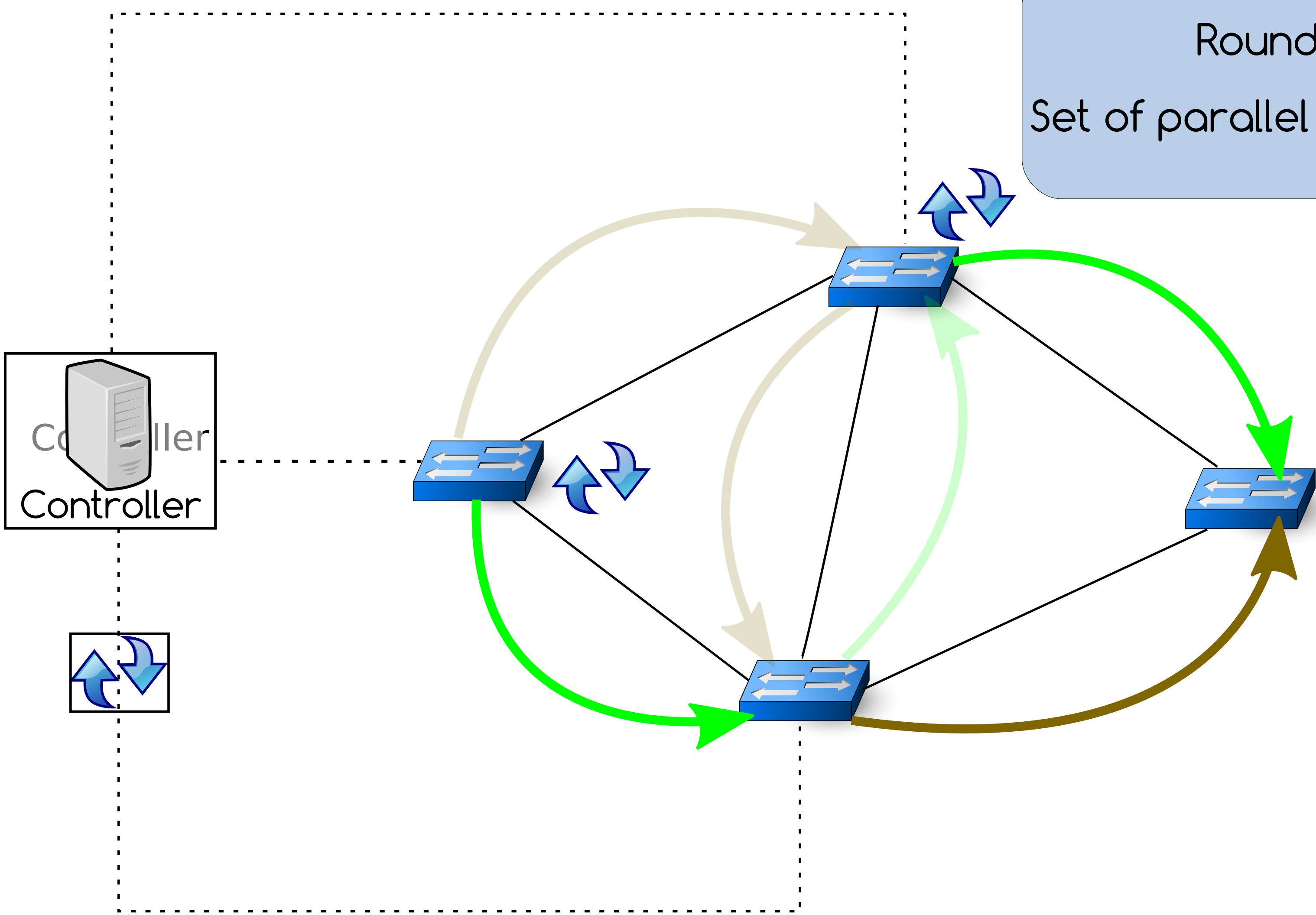
Asynchronous Updates: Round model



Round:

Set of parallel updates.

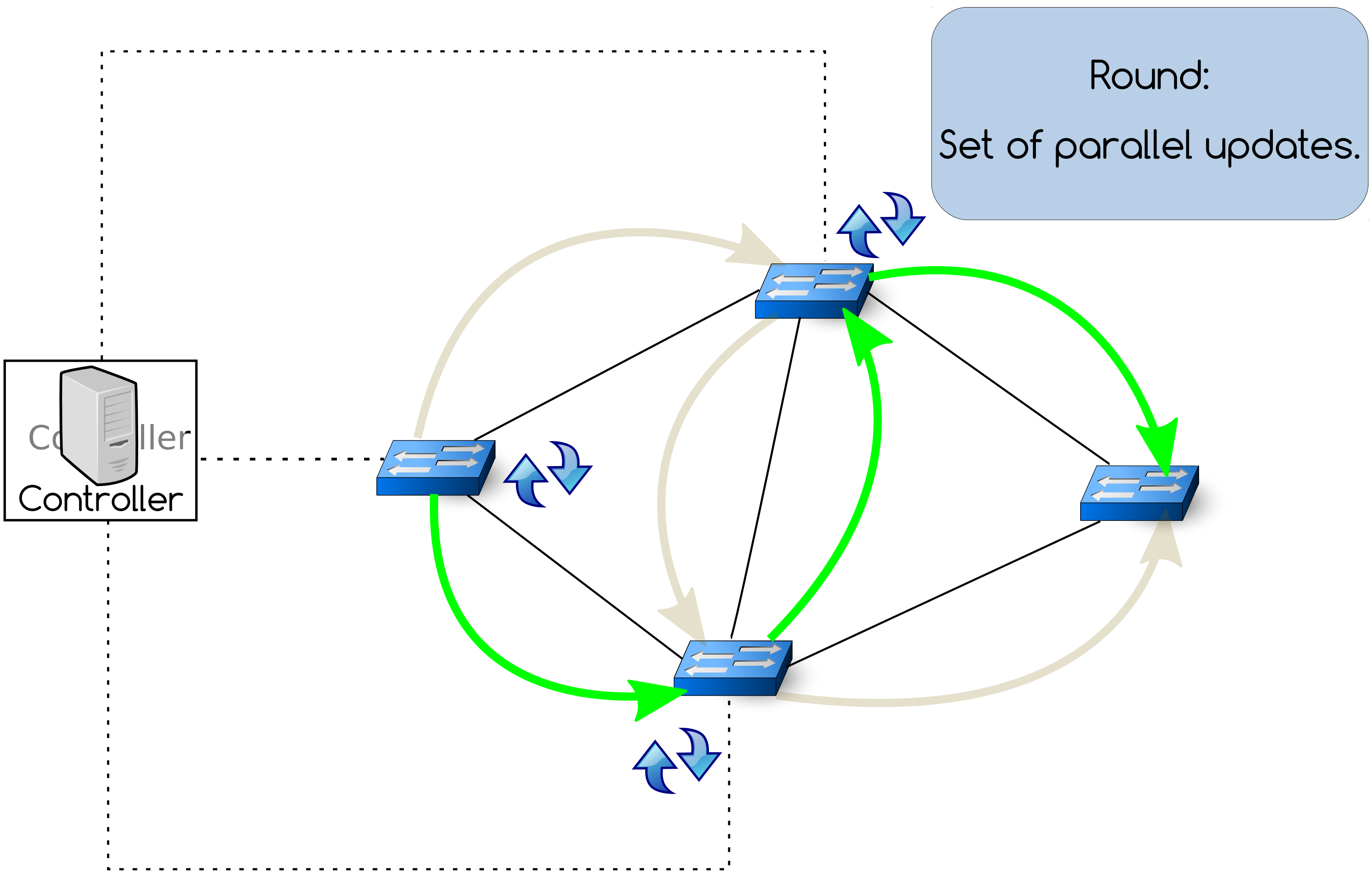
Asynchronous Updates: Round model



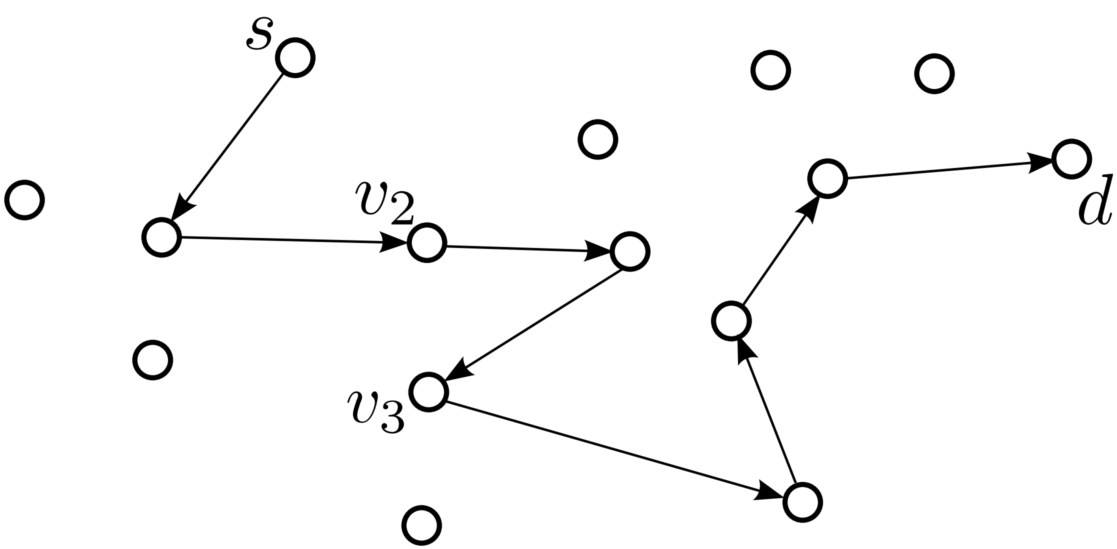
Round:

Set of parallel updates.

Asynchronous Updates: Round model



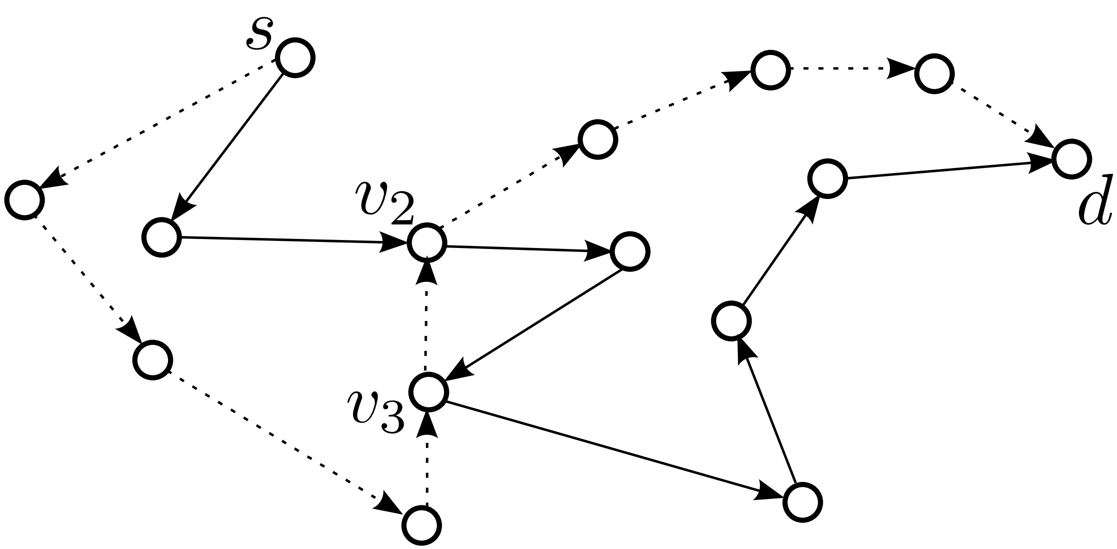
Model Representation

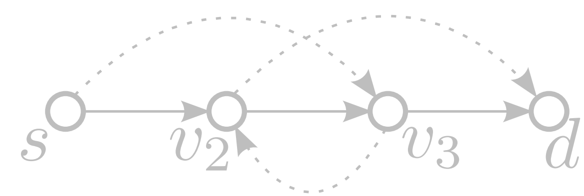


- Solid lines = current path
- Dashed lines = new path (Flow-specific path)

Model Representation

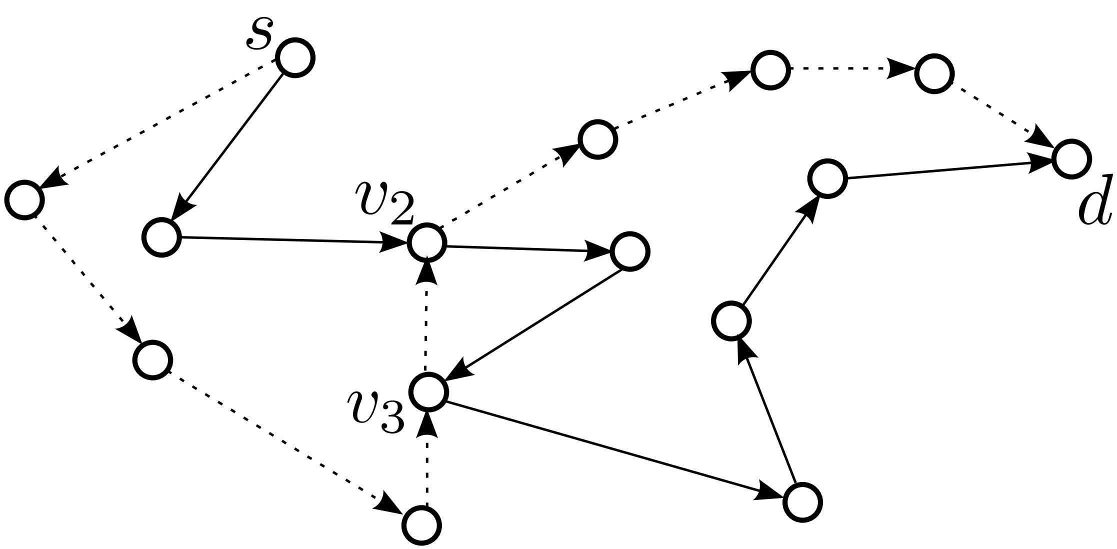
| Solid lines = current path
┆ Dashed lines = new path
(Flow-specific path)

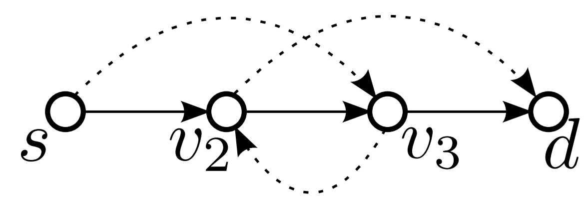




Model Representation

- Solid lines = current path
- Dashed lines = new path (Flow-specific path)



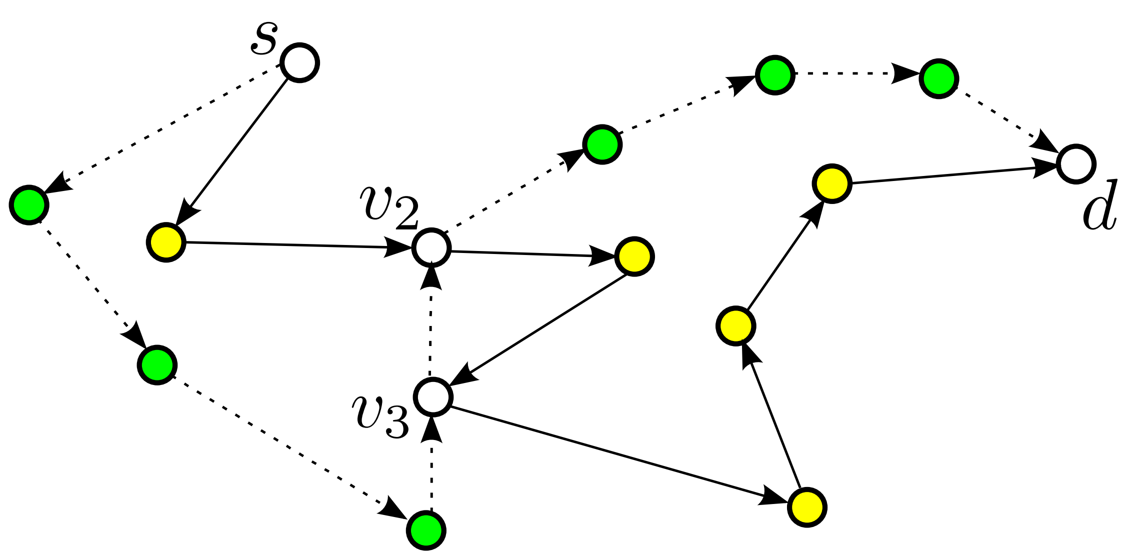


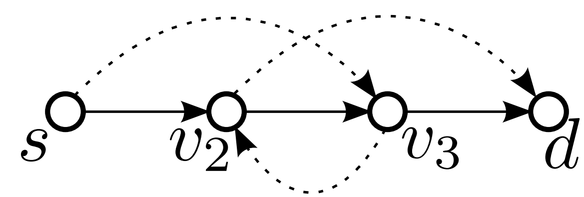
Model Representation

Solid lines = current path

Dashed lines = new path

(Flow-specific path)

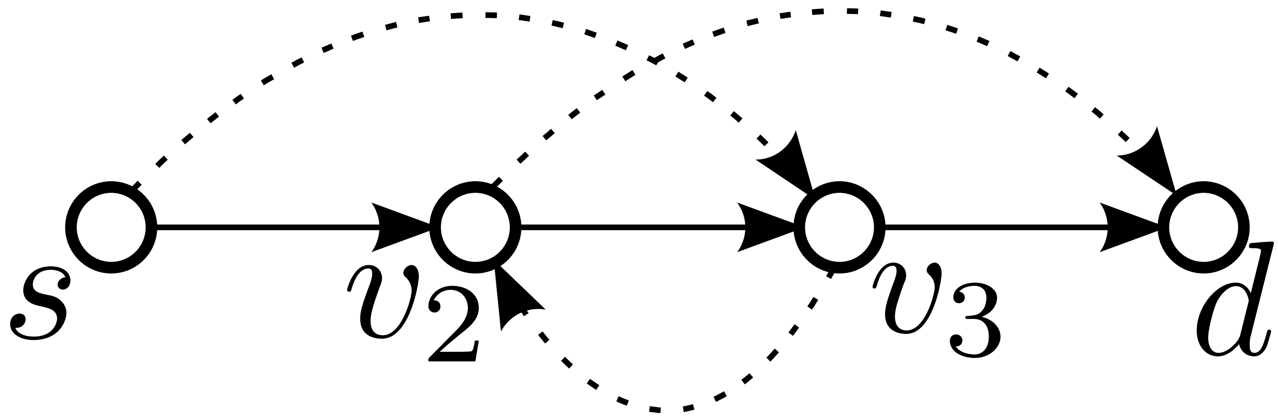




Safe to be updated

Safe to be left untouched

Model Representation

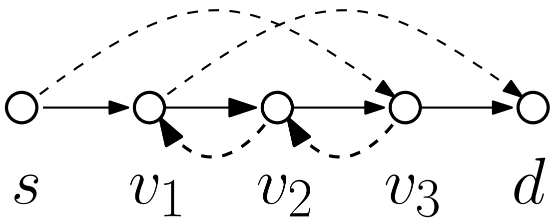


| Solid lines = current path ⁞ Dashed lines = new path

Consistency Properties

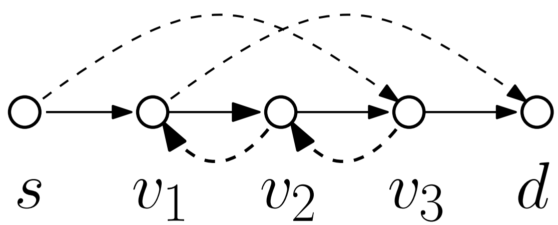
Property: Strong Loop Freedom (SLF)

State

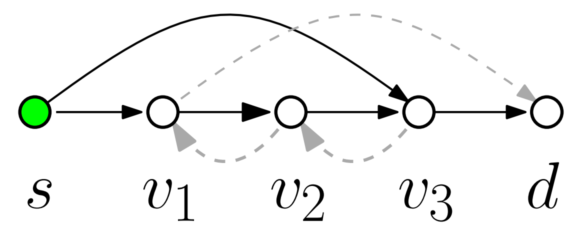


Property: Strong Loop Freedom (SLF)

State

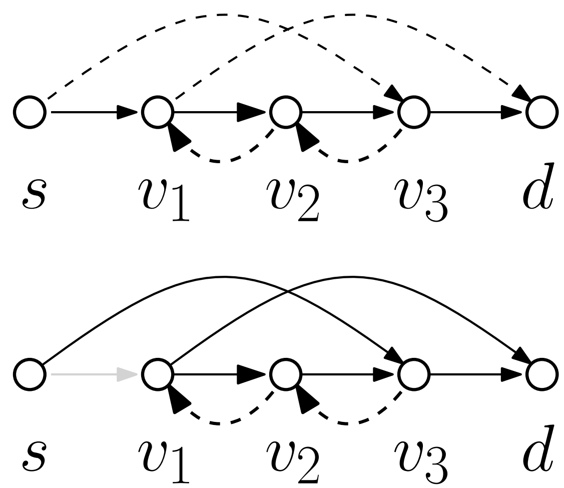


Temporary Forwarding Graph

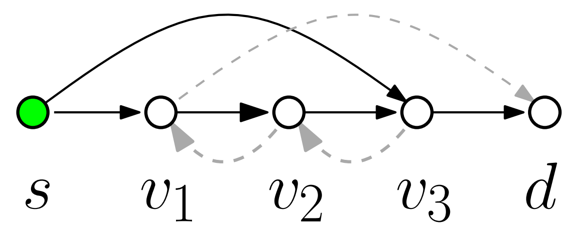


Property: Strong Loop Freedom (SLF)

State

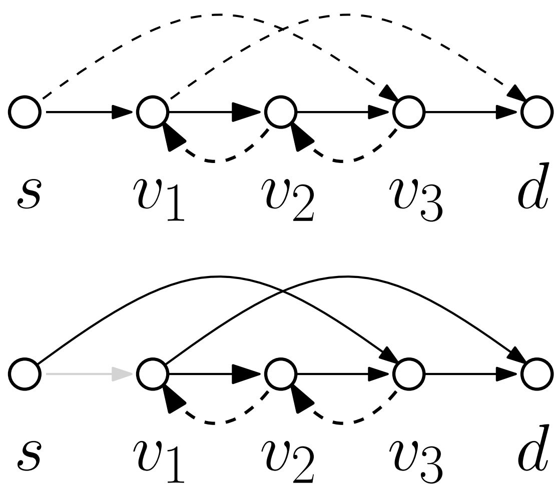


Temporary Forwarding Graph

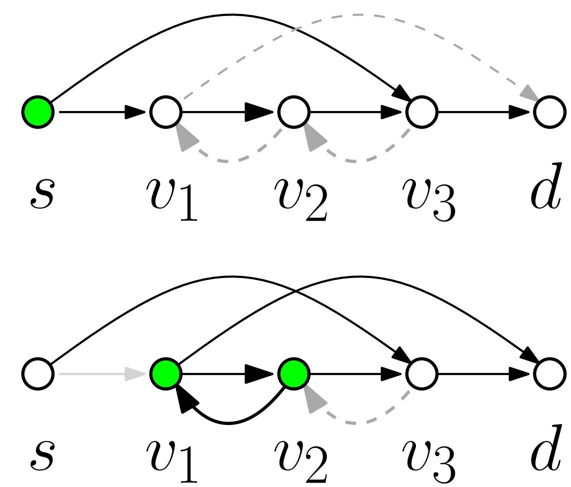


Property: Strong Loop Freedom (SLF)

State

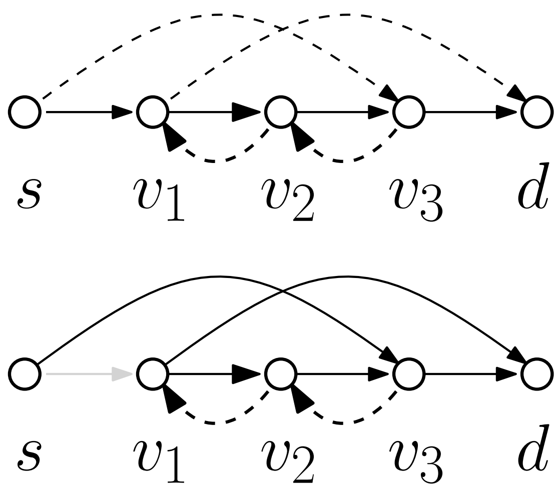


Temporary Forwarding Graph

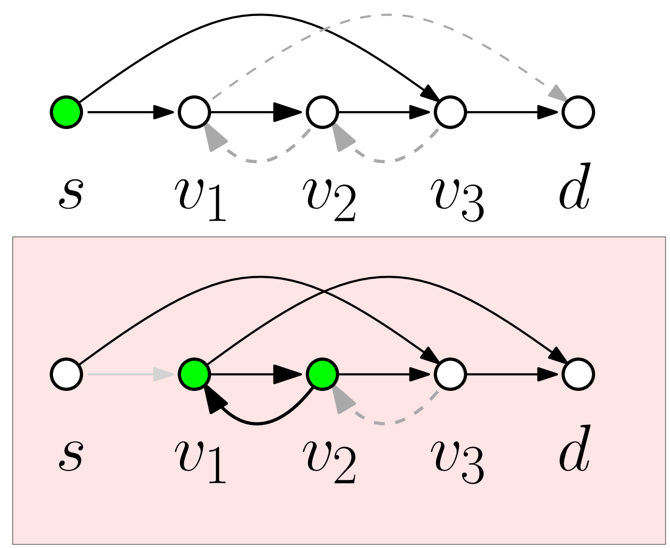


Property: Strong Loop Freedom (SLF)

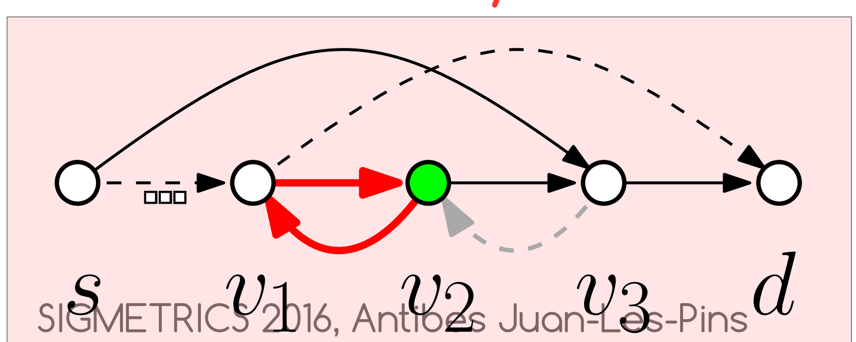
State



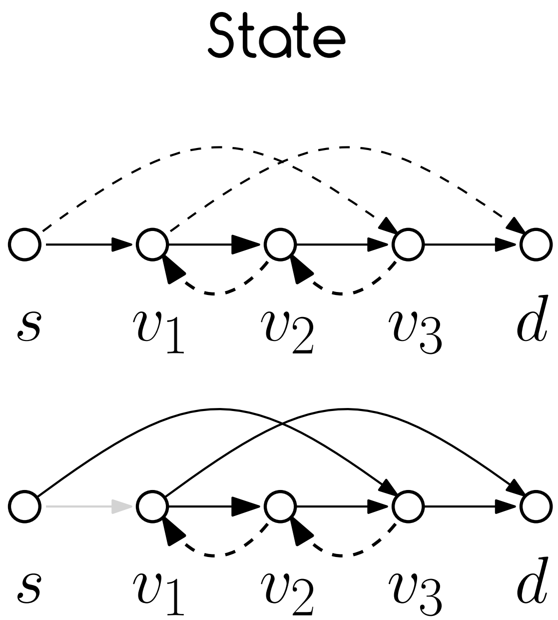
Temporary Forwarding Graph

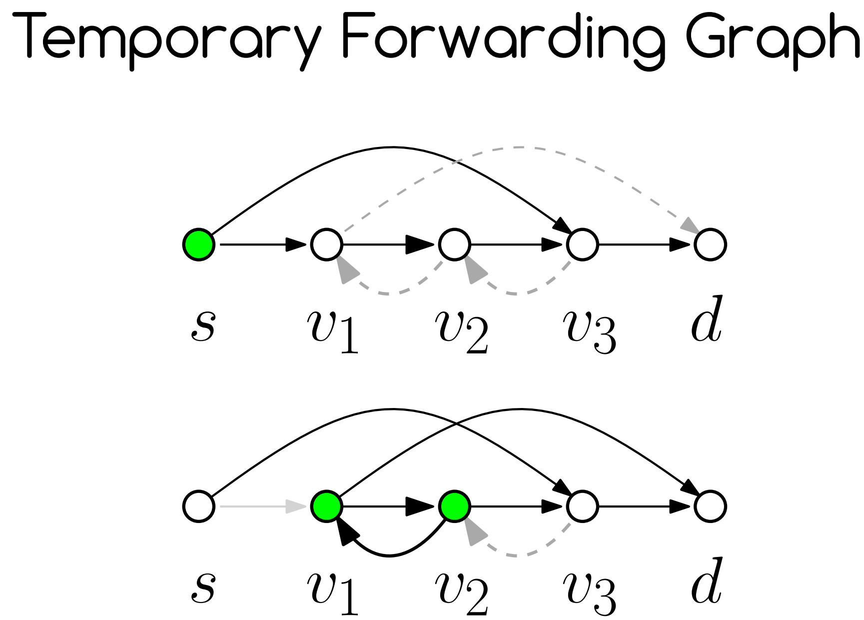


Temporary forwarding graph
– i.e. the union of previously and newly enabled edges –
does not contain any directed loop.



Property: Strong Loop Freedom (SLF)

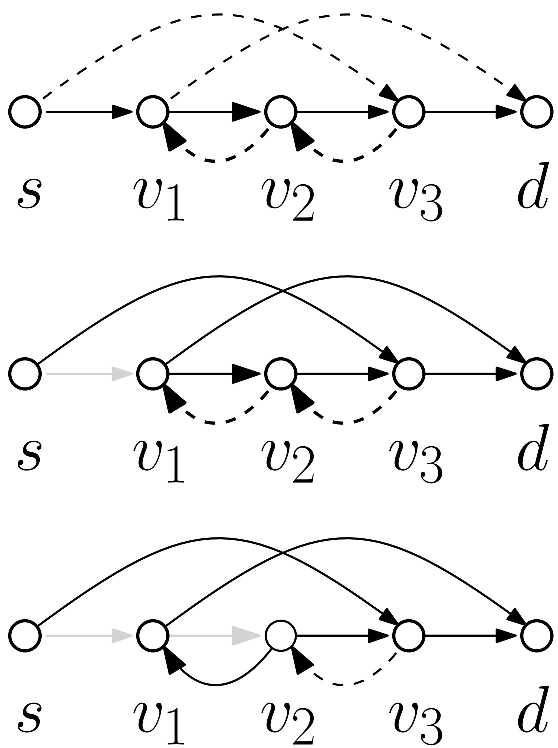




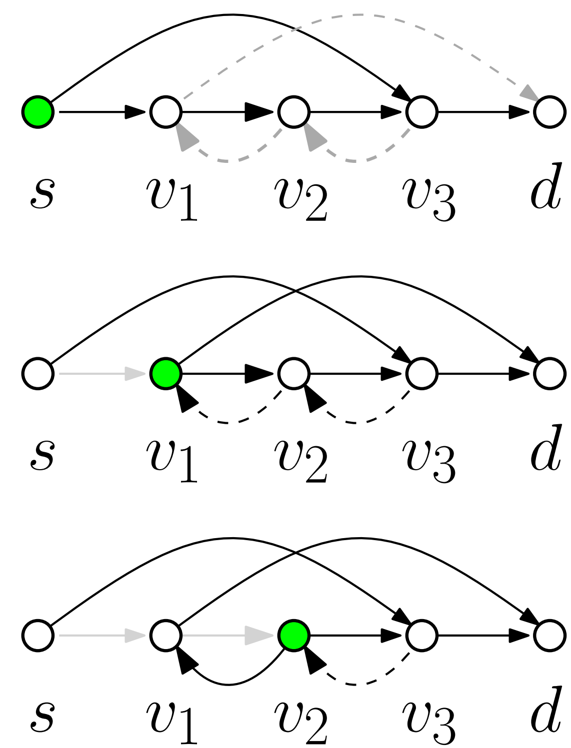
Temporary forwarding graph
– i.e. the union of previously and newly enabled edges –
does not contain any directed loop.

Property: Strong Loop Freedom (SLF)

State

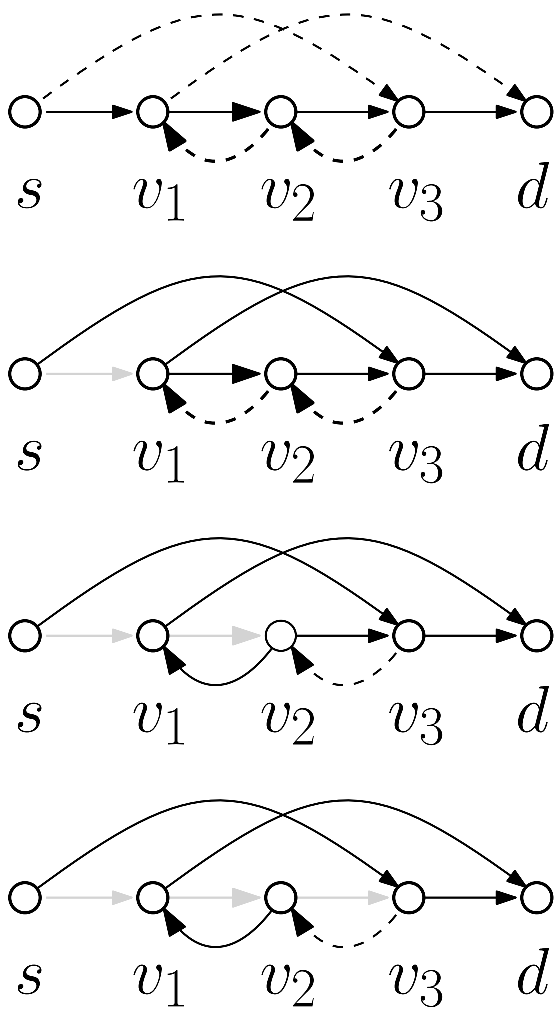


Temporary Forwarding Graph

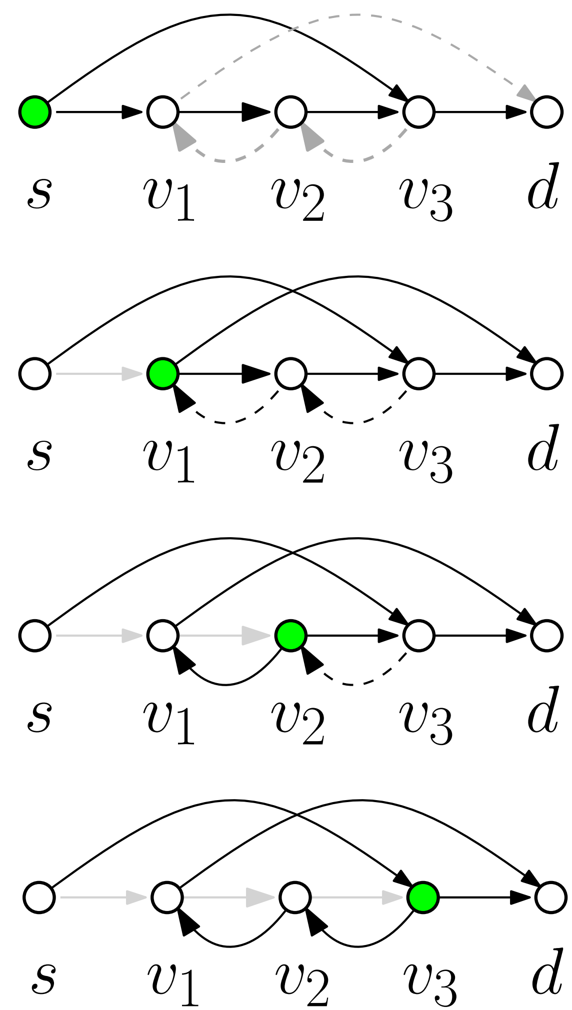


Property: Strong Loop Freedom (SLF)

State

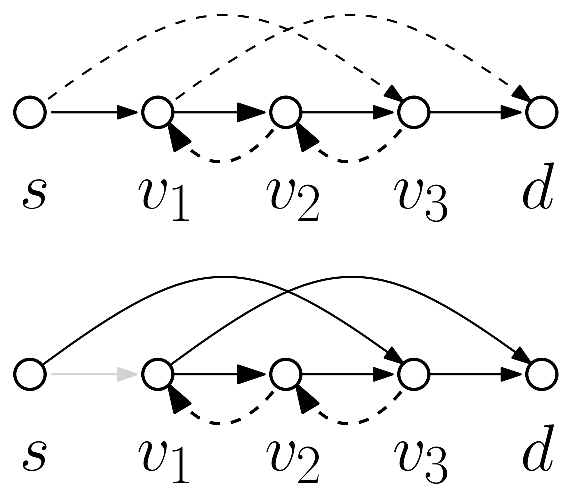


Temporary Forwarding Graph

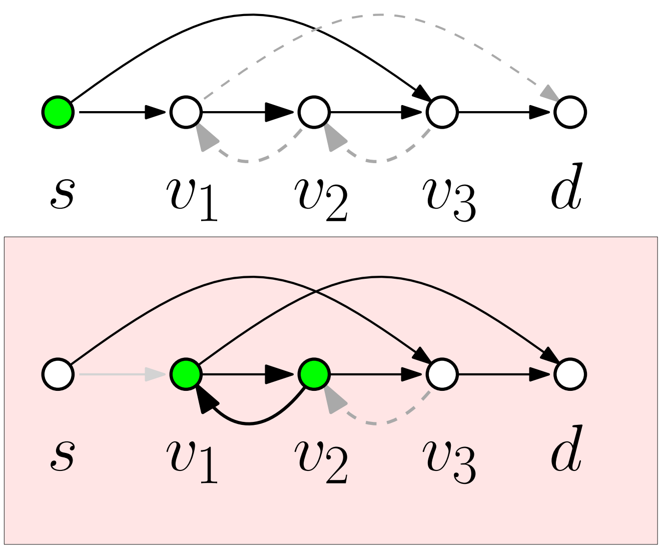


Property: Relaxed Loop Freedom (RLF)

State

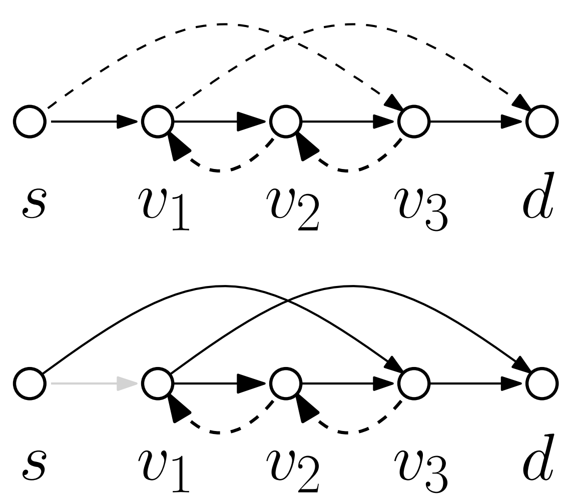


Temporary Forwarding Graph

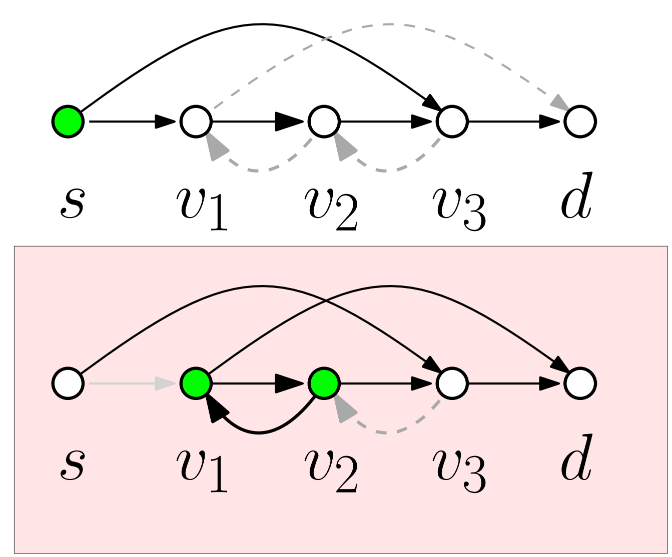


Property: Relaxed Loop Freedom (RLF)

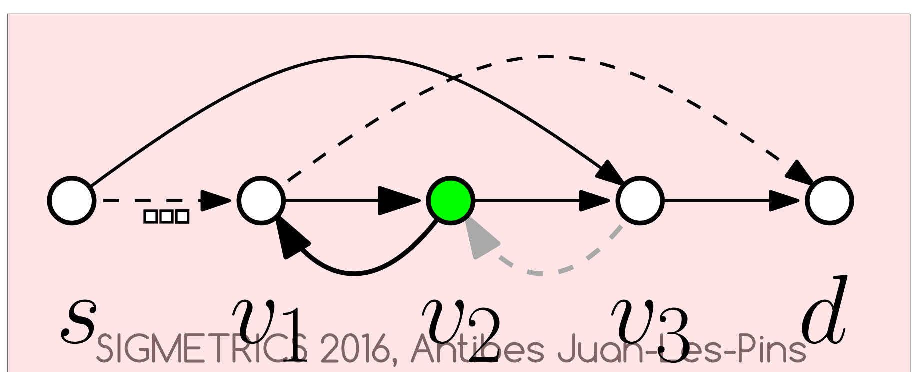
State



Temporary Forwarding Graph

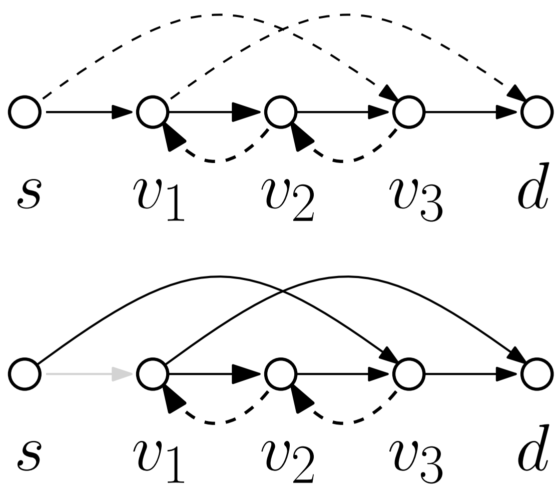


Connected component of the temporary forwarding graph containing the source does not contain directed loops.

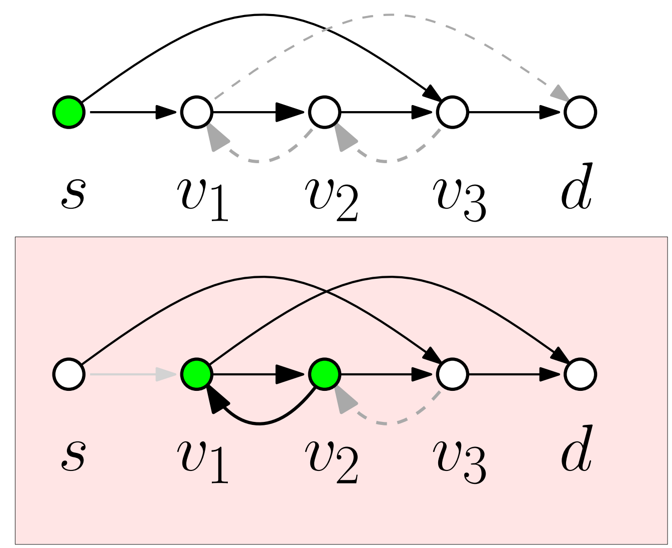


Property: Relaxed Loop Freedom (RLF)

State

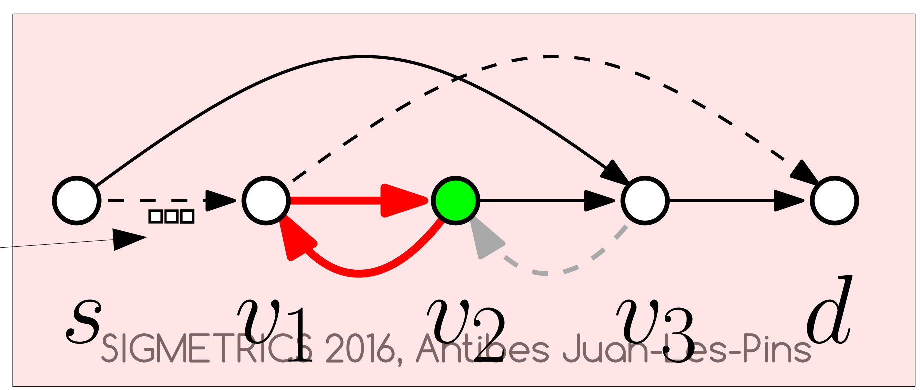


Temporary Forwarding Graph

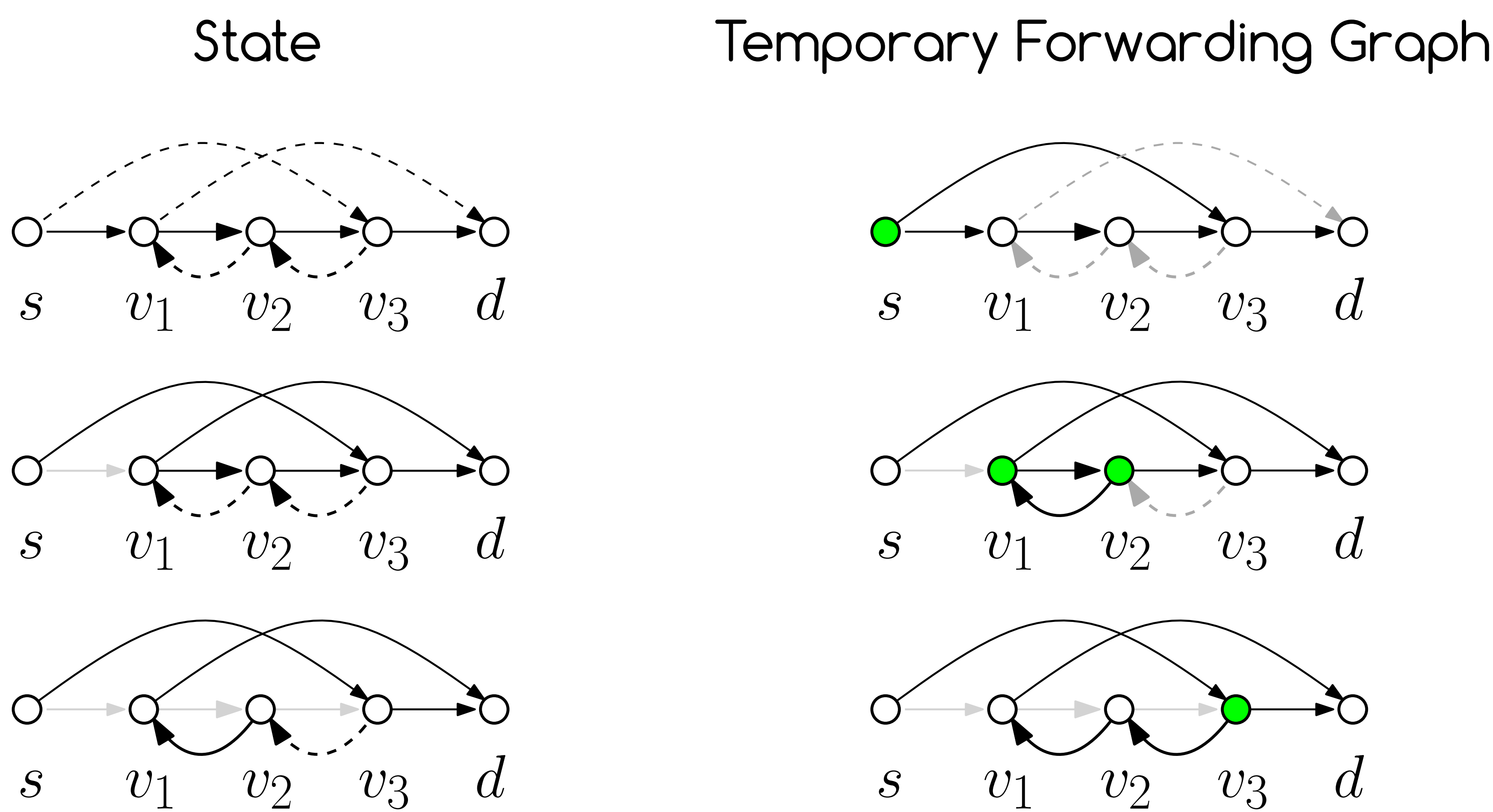


Connected component of the temporary forwarding graph containing the source does not contain directed loops.

Finitely many packets

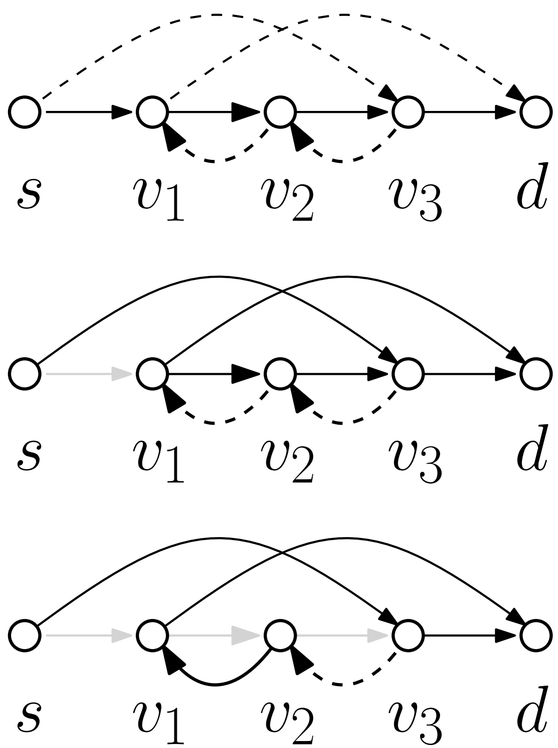


Property: Relaxed Loop Freedom (RLF)

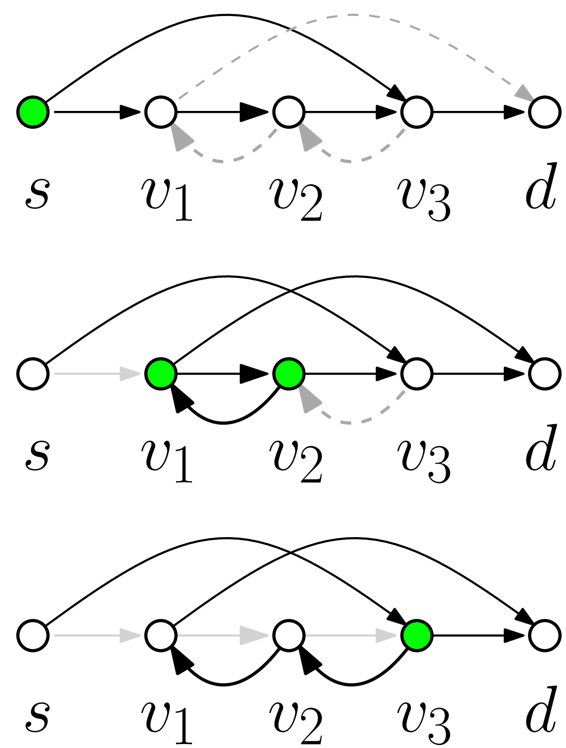


Property: Relaxed Loop Freedom (RLF)

State



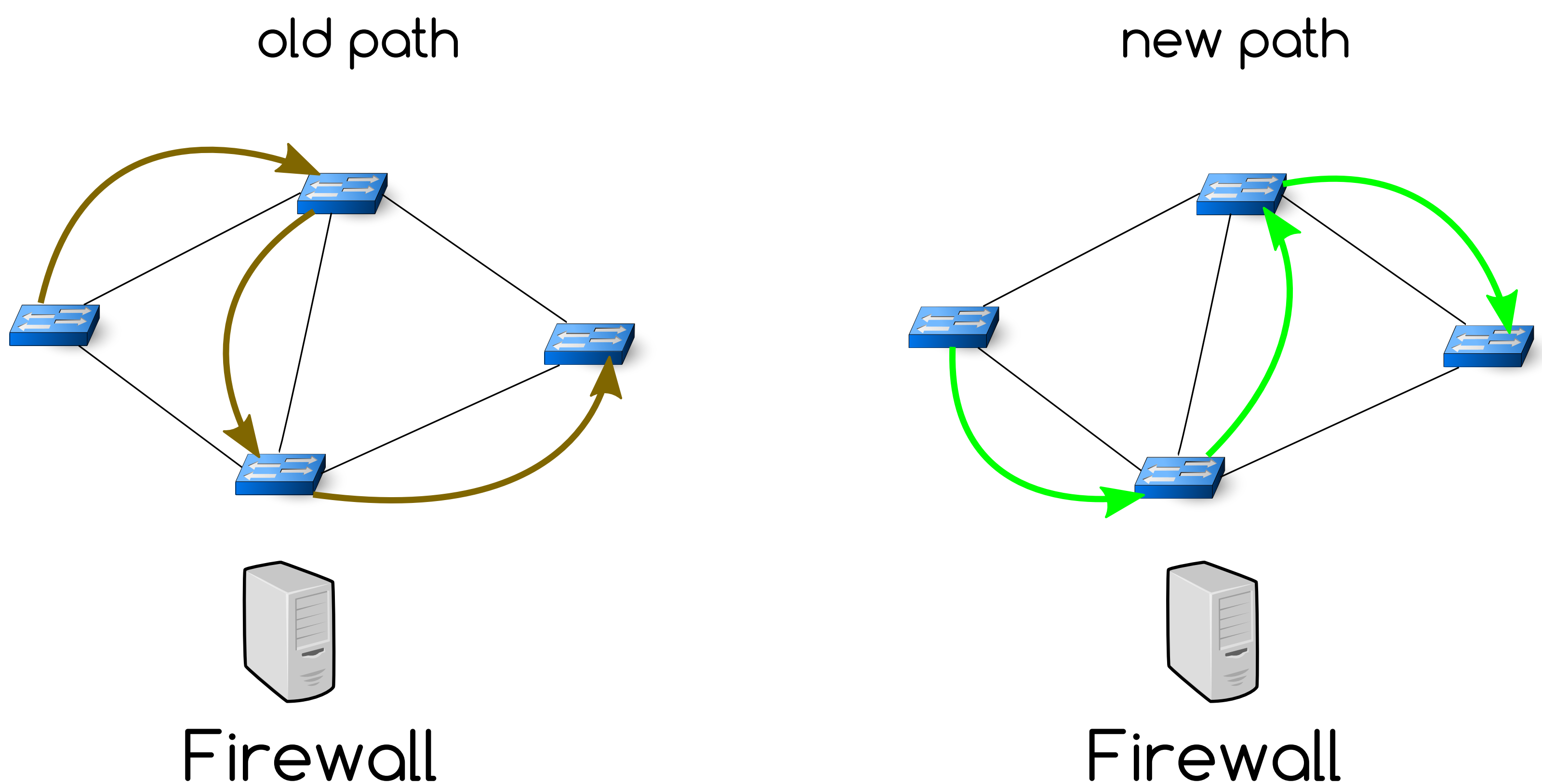
Temporary Forwarding Graph



Observation: RLF requires one round less than SLF.

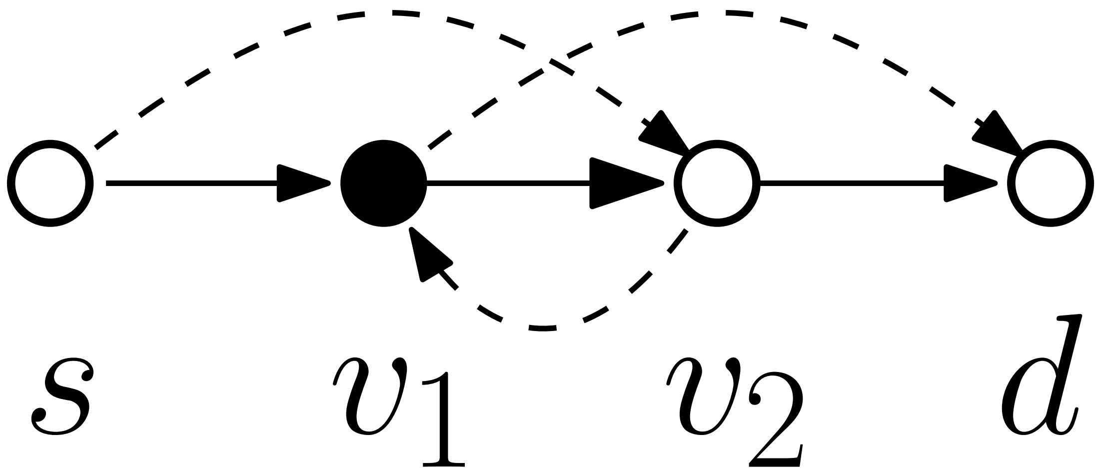
Property: Waypoint Enforcement (WPE)

Increasing number of middleboxes [Sherry et al., SIGCOMM '12]



Property: Waypoint Enforcement (WPE)

'Waypoint (e.g. firewall) may never be bypassed.'

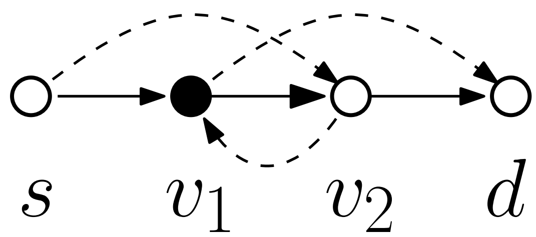


Solid lines = current path

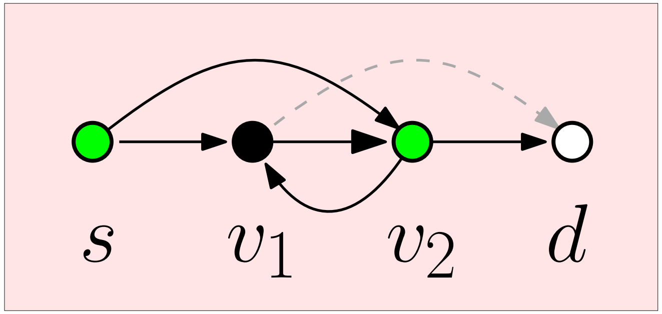
Dashed lines = new path

Property: Waypoint Enforcement (WPE)

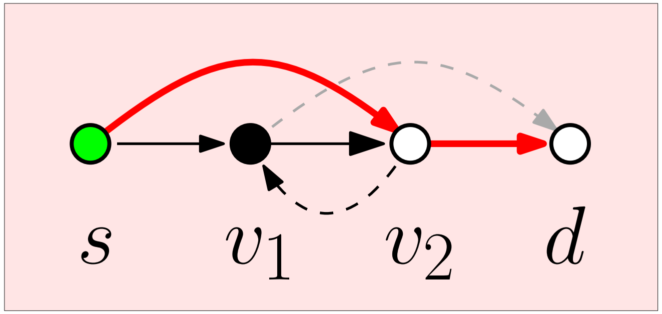
State



Temporary Forwarding Graph

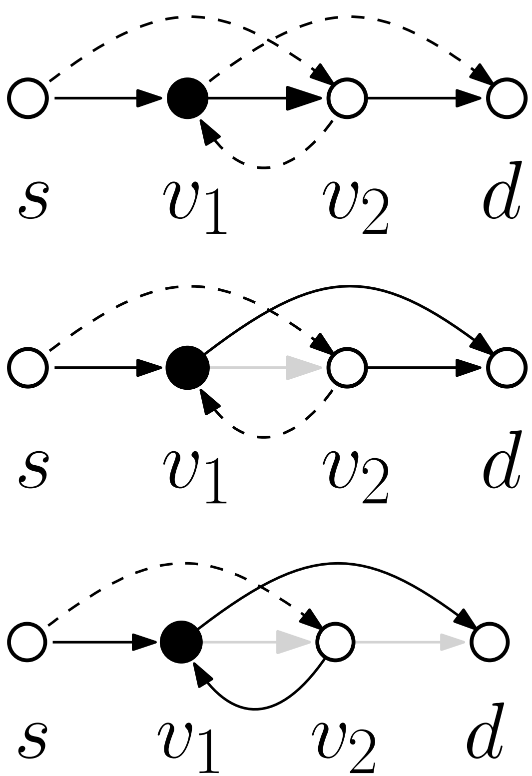


There may not exist a path bypassing the waypoint in the Temporary Forwarding Graph.

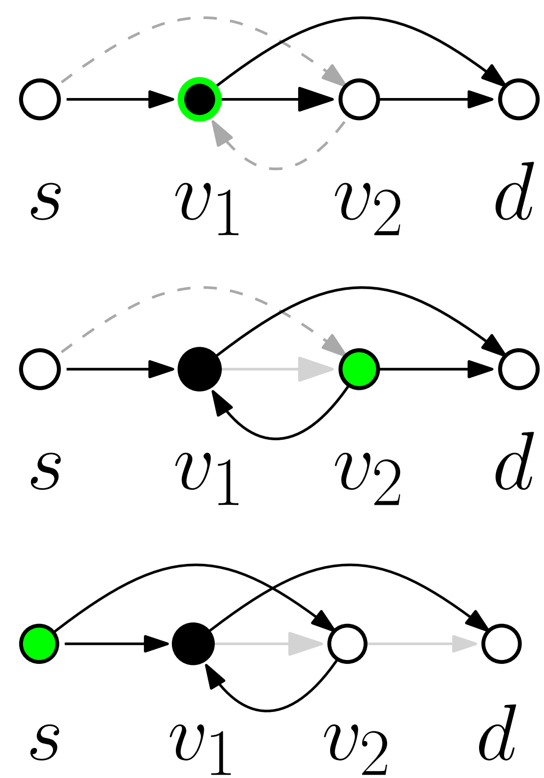


Property: Waypoint Enforcement (WPE)

State



Temporary Forwarding Graph



Overview

Task: Minimize overall update time, while

- ensuring Loop Freedom (LF)
- ensuring Waypoint Enforcement (WPE)

Theory

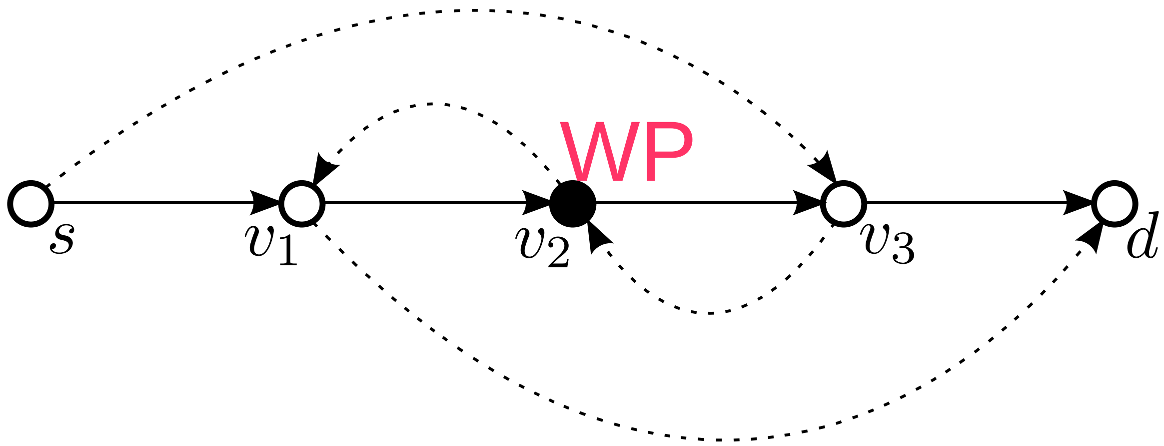
- LF + WPE may conflict
- Deciding LF + WPE is NP-hard
- other 'negative' results

Practice

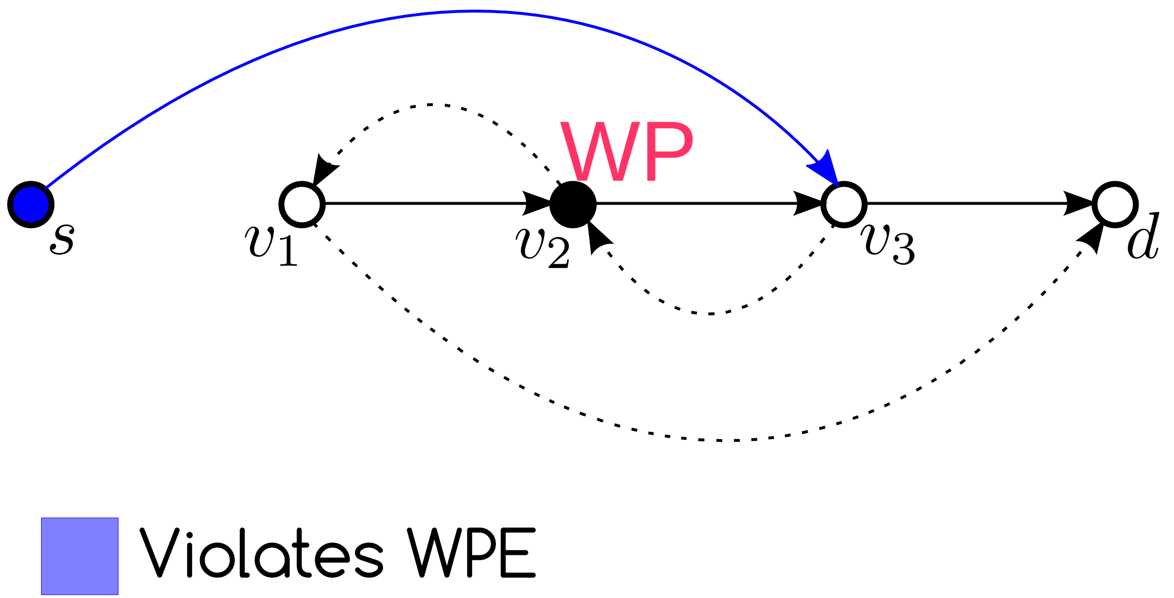
- Mixed-Integer Programming Formulations
- Qualitative and Quantitative Analysis

Theory:
LF and WPE may conflict

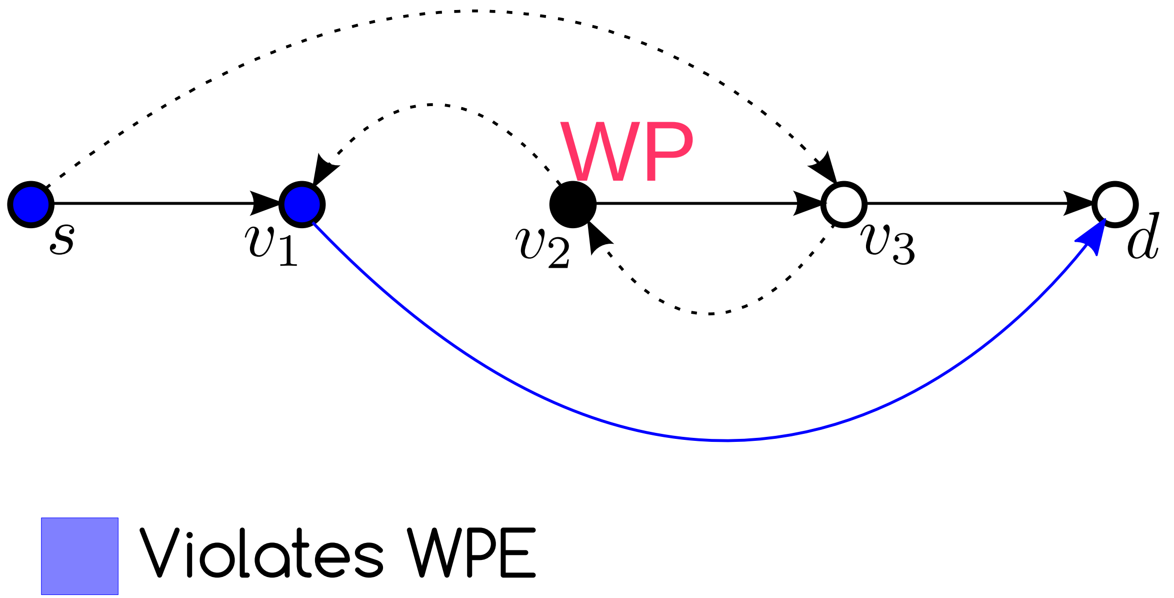
LF and WPE may Conflict



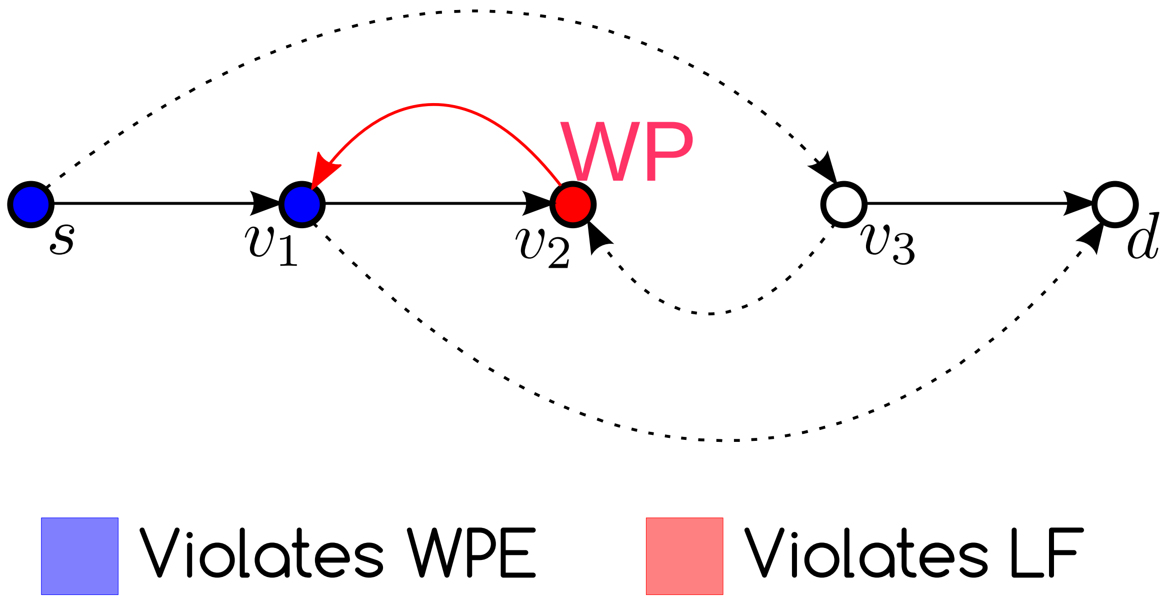
LF and WPE may Conflict



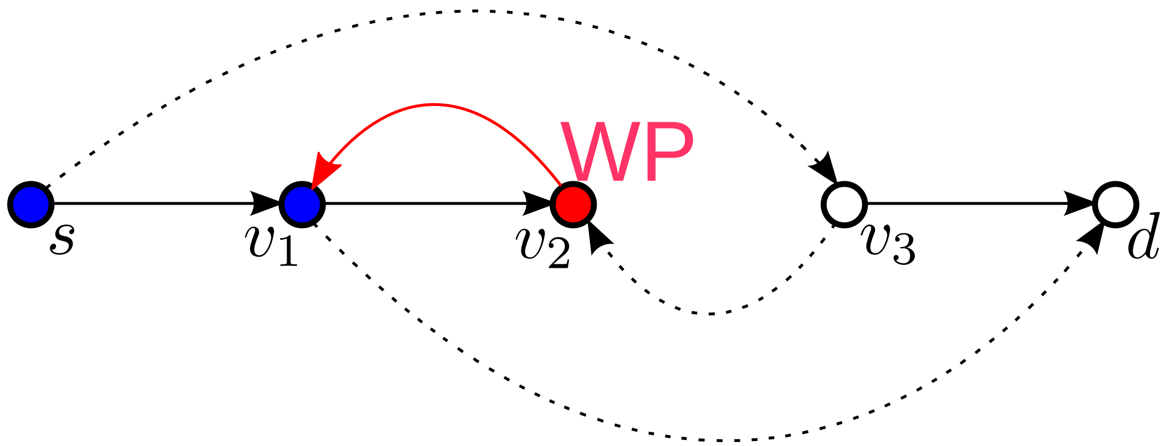
LF and WPE may Conflict



LF and WPE may Conflict

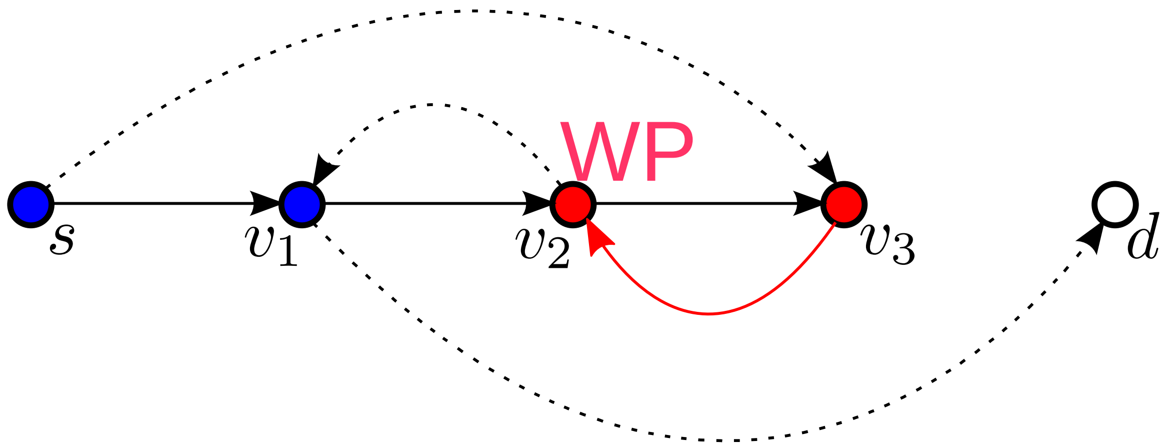


LF and WPE may Conflict



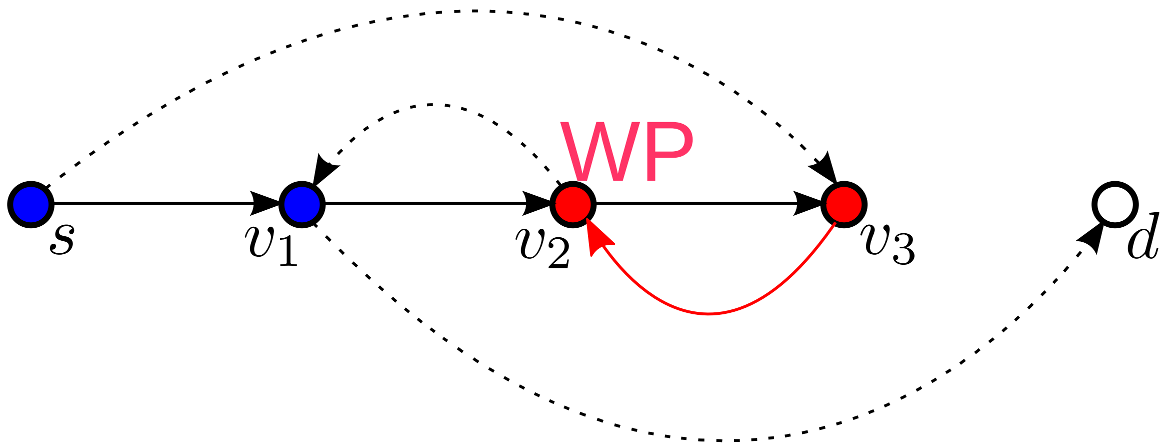
Violates WPE   Violates LF

LF and WPE may Conflict



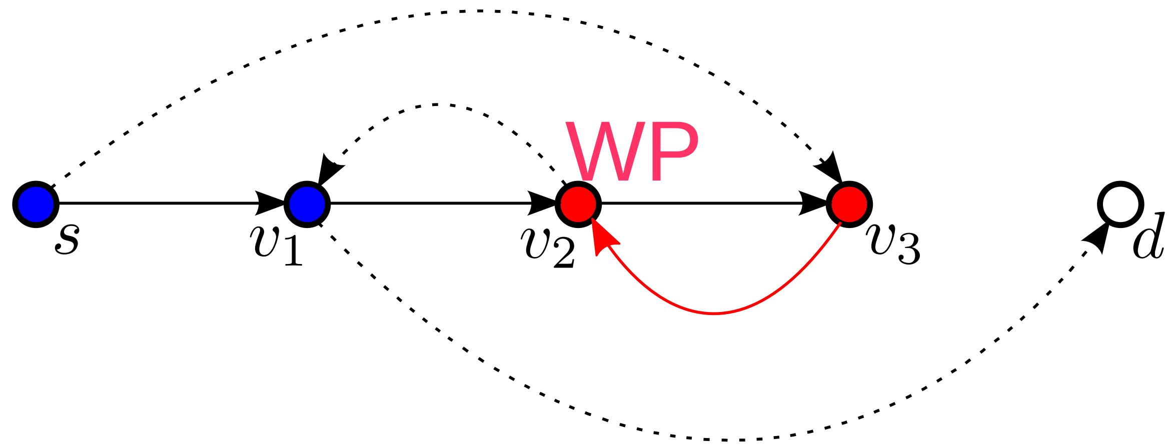
Violates WPE   Violates LF

LF and WPE may Conflict



Some update problems are unsolvable when considering LF and WPE.

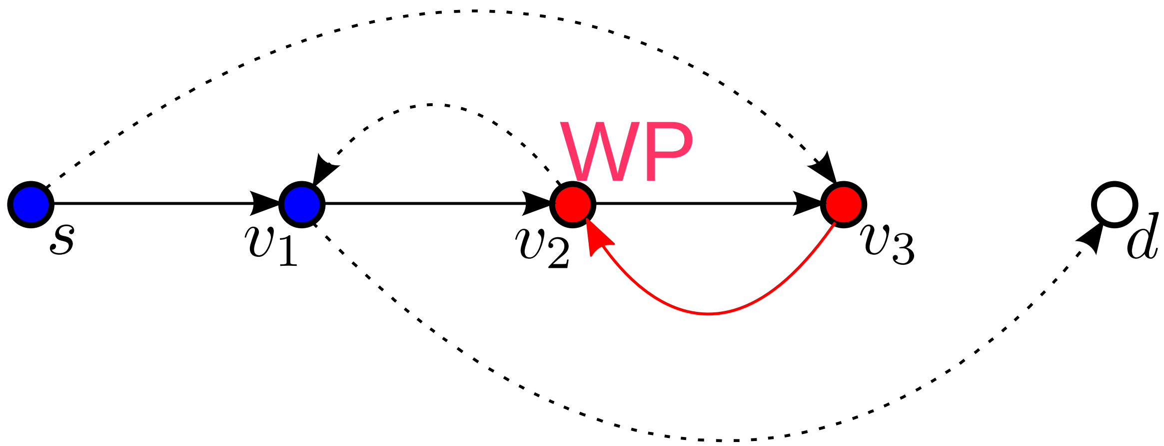
LF and WPE may Conflict



Some update problems are unsolvable when considering LF and WPE.

Independent of whether RLF or SLF is considered.

LF and WPE may Conflict



Some update problems are unsolvable when considering LF and WPE.

Can we determine these cases easily?

Theory:
Deciding whether an Update Schedule exists is NP-hard

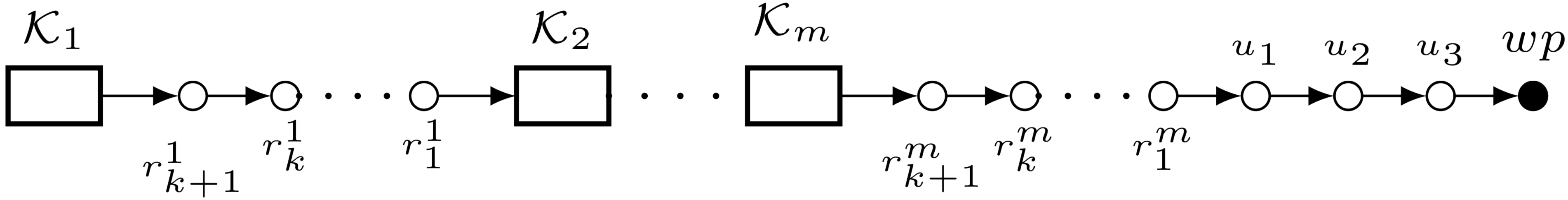
Deciding existence of Schedule is NP-hard

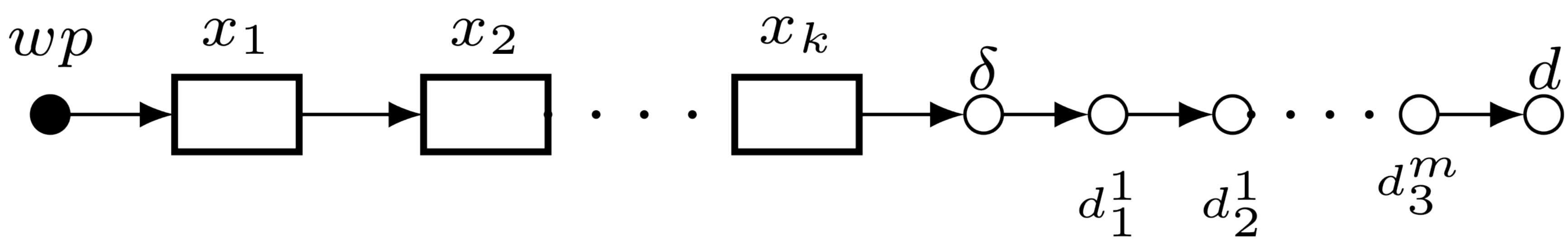
- Proof by 3-SAT reduction
  - Given a 3-SAT formula we construct a network update instance and show that there exists an update schedule iff. the formula is satisfiable.

Deciding existence of Schedule is NP-hard

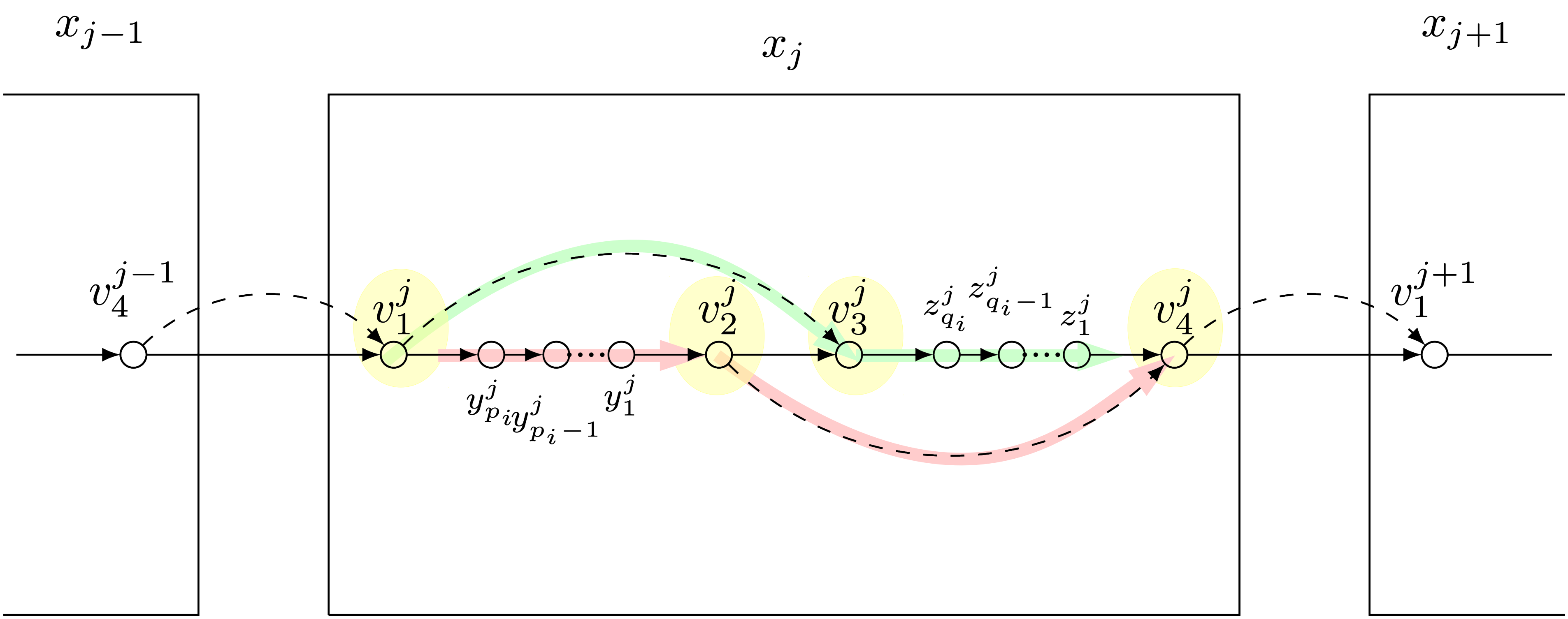
- Proof by 3-SAT reduction
  - Given a 3-SAT formula we construct a network update instance and show that there exists an update schedule iff. the formula is satisfiable.
  - 3-SAT Clause $\mathcal{K}_1 \wedge \mathcal{K}_2 \wedge \ldots \wedge \mathcal{K}_m$ over Variables $x_{1,} x_{2,} \ldots, x_k$
  - **Here: we only sketch the idea.**

Construction of 3-SAT Reduction: Outline



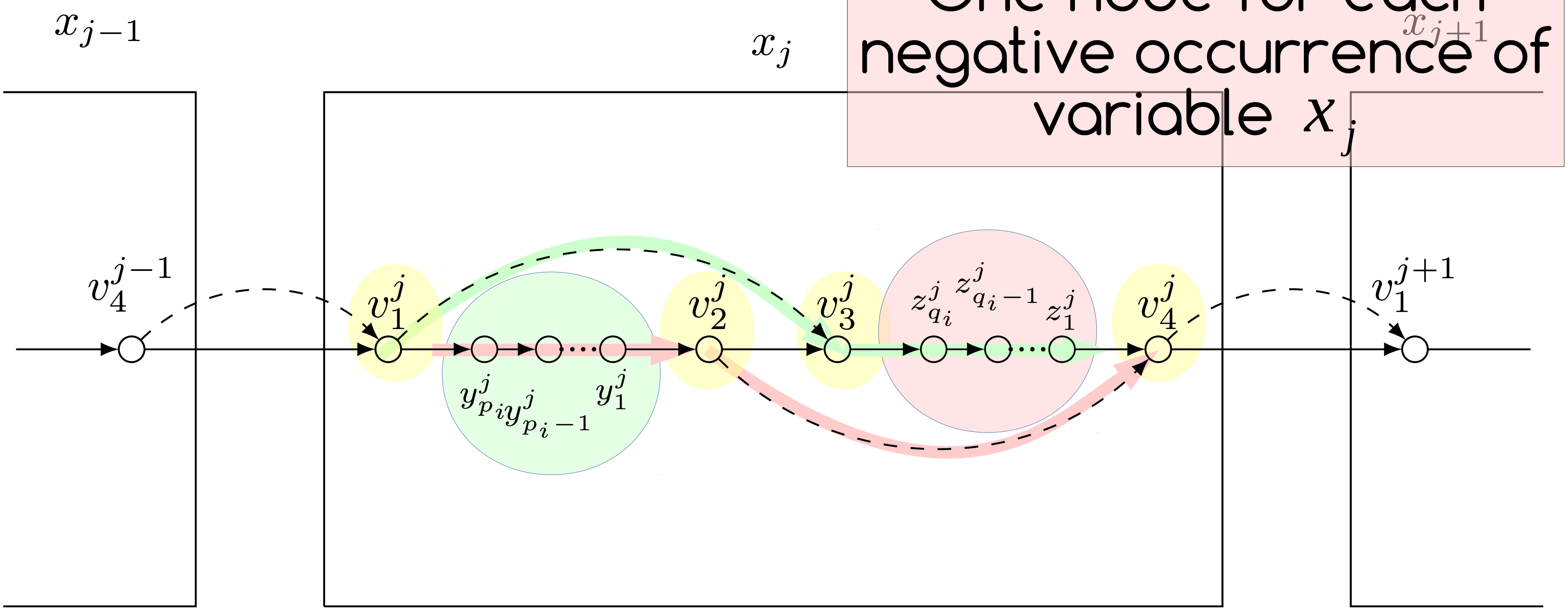


Construction of 3-SAT Reduction: Variable Gadgets



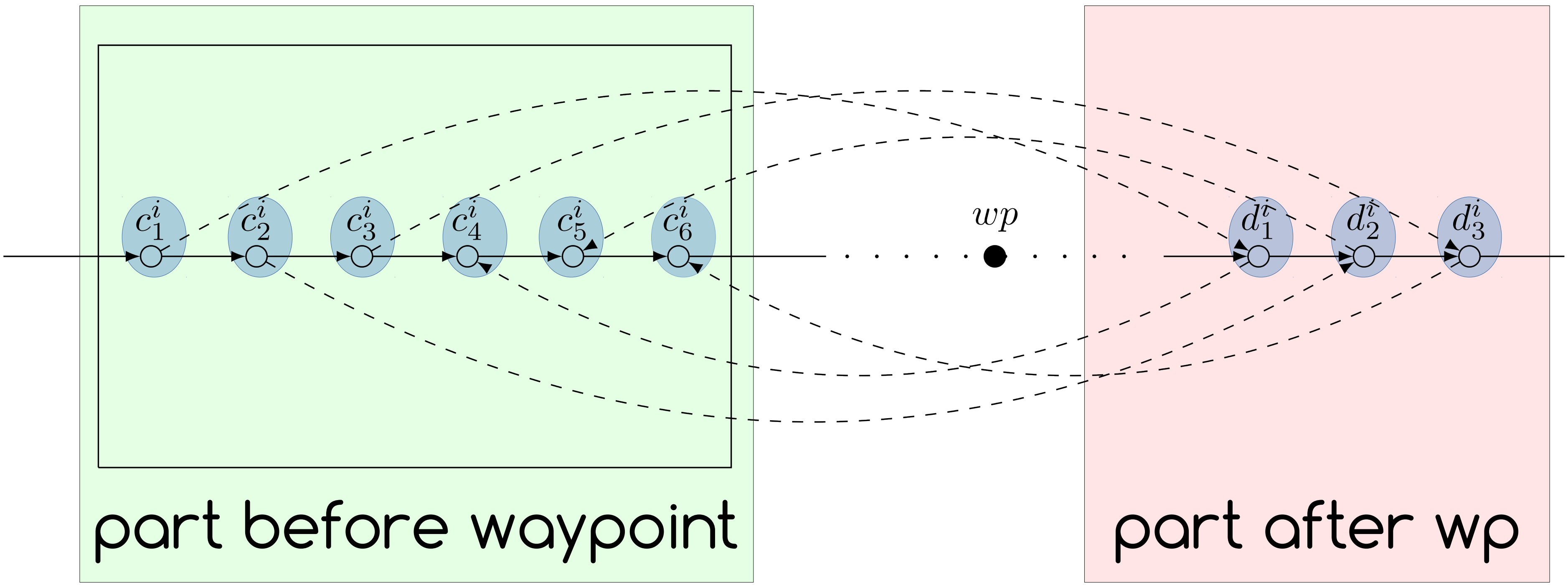
Construction of 3-SAT Reduction: Variable Gadgets

One node for each negative occurrence of variable x_j

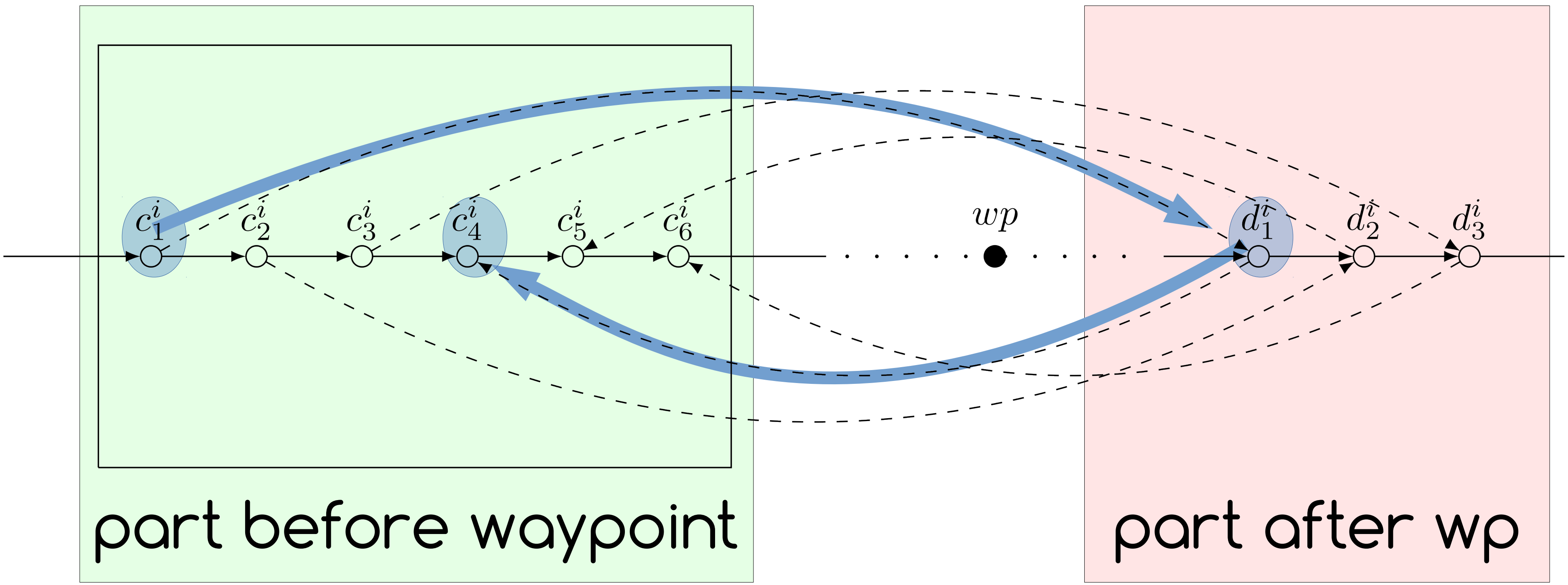


One node for each positive occurrence of variable x_j

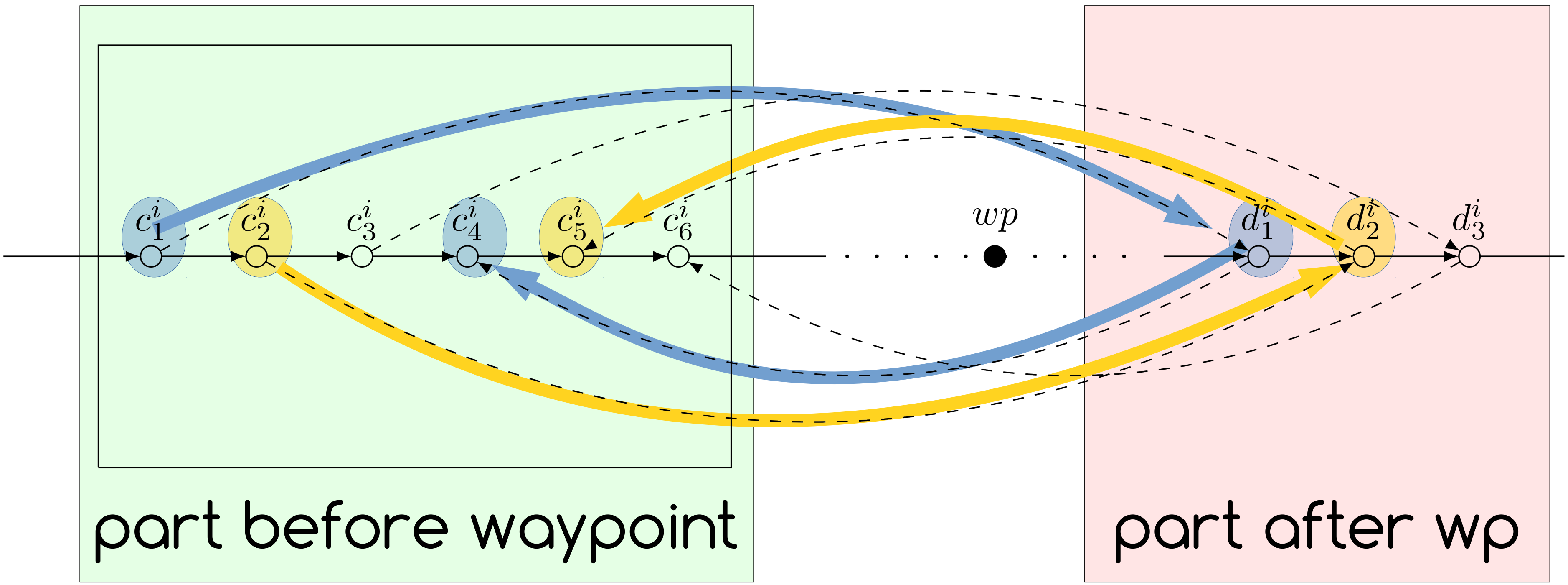
Construction of 3-SAT Reduction: Clause Gadgets



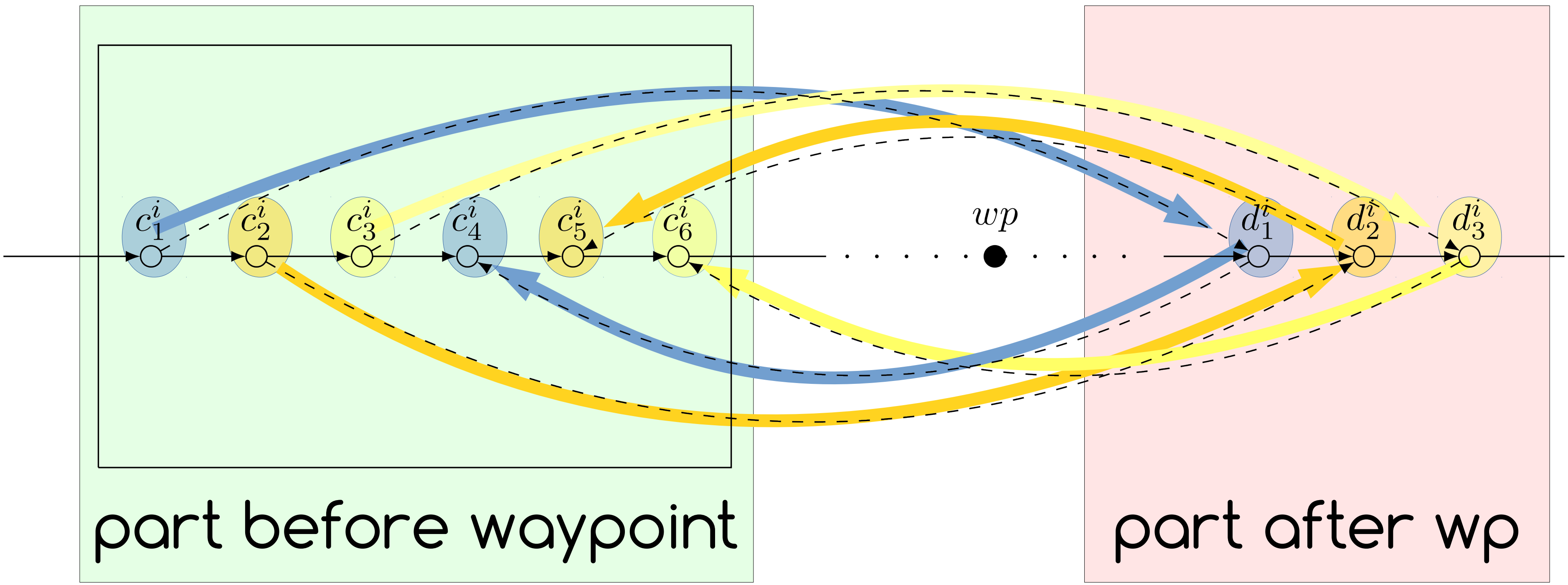
Construction of 3-SAT Reduction: Clause Gadgets



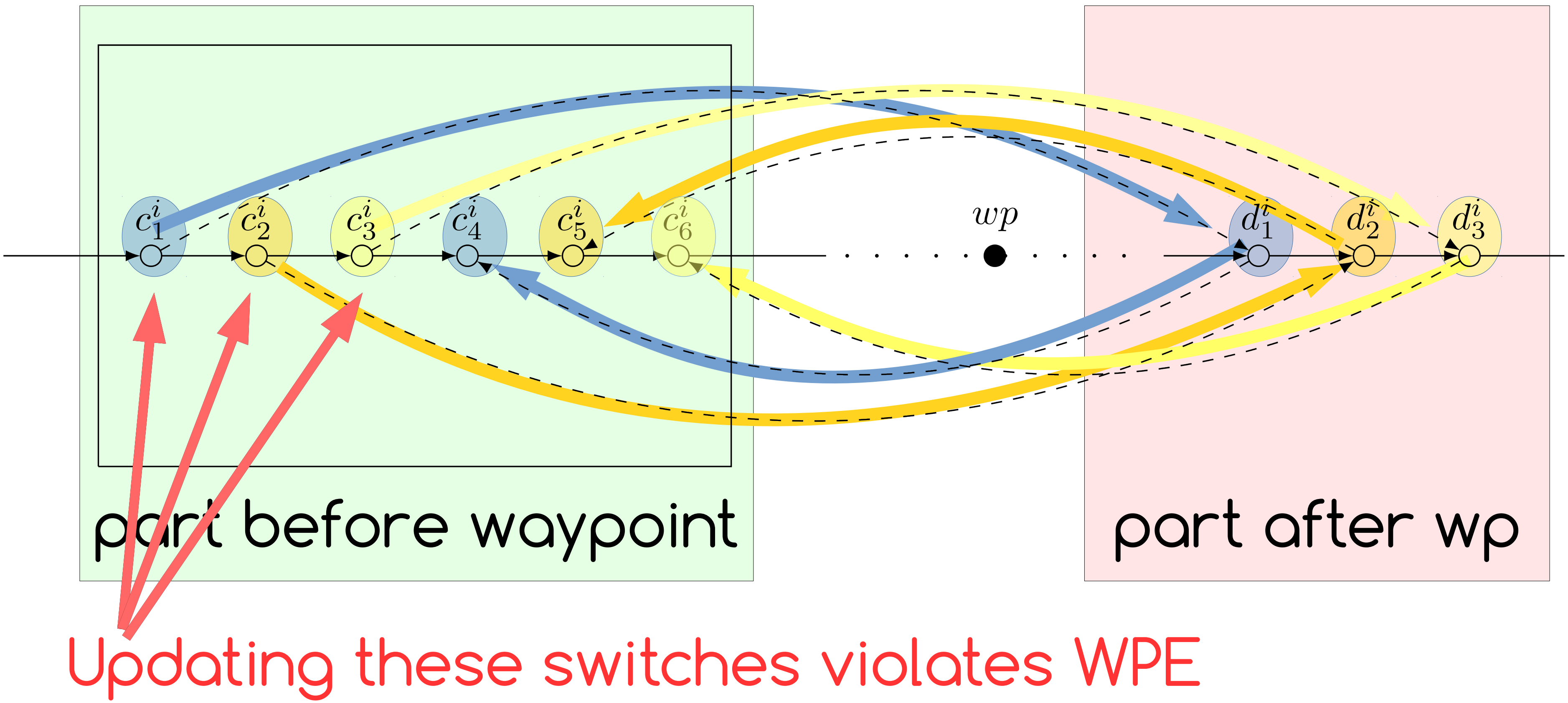
Construction of 3-SAT Reduction: Clause Gadgets



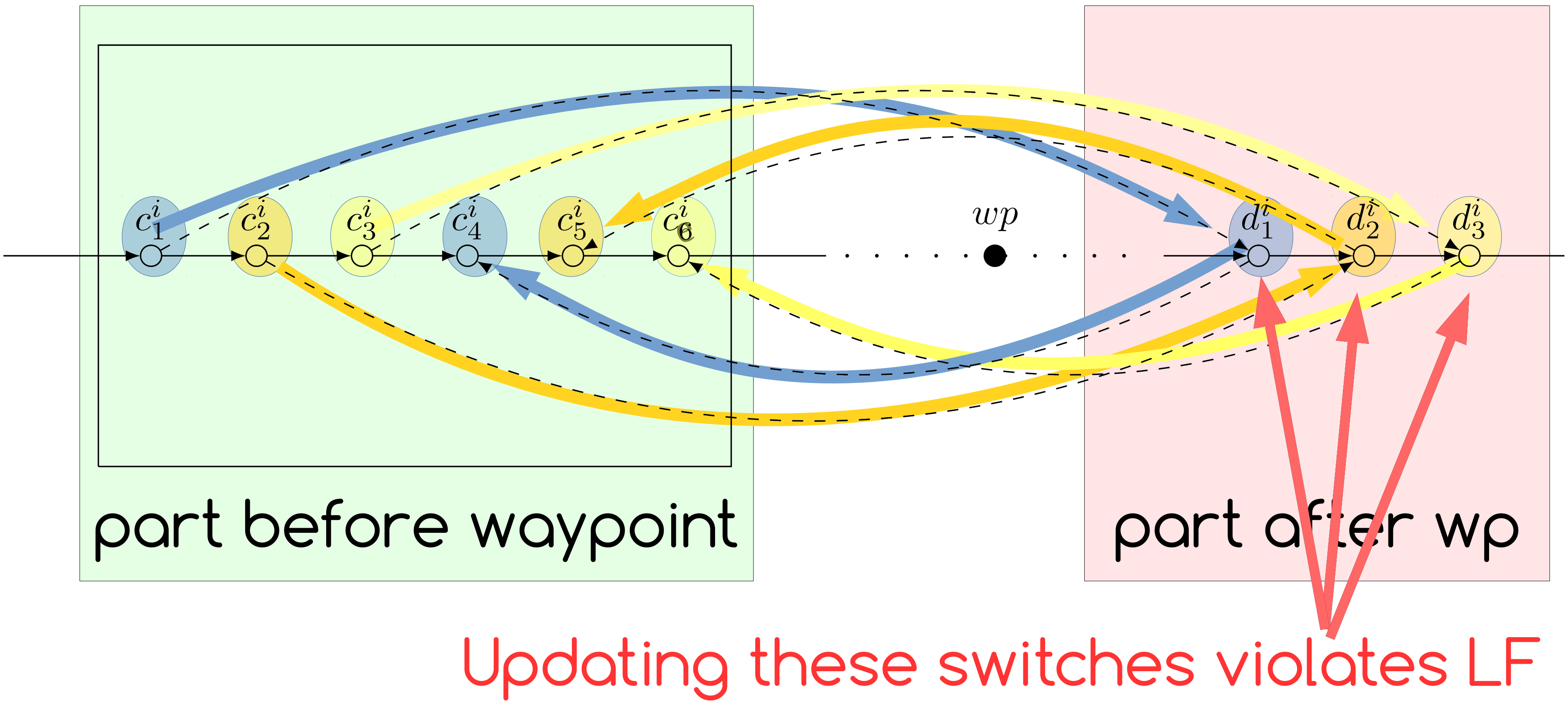
Construction of 3-SAT Reduction: Clause Gadgets



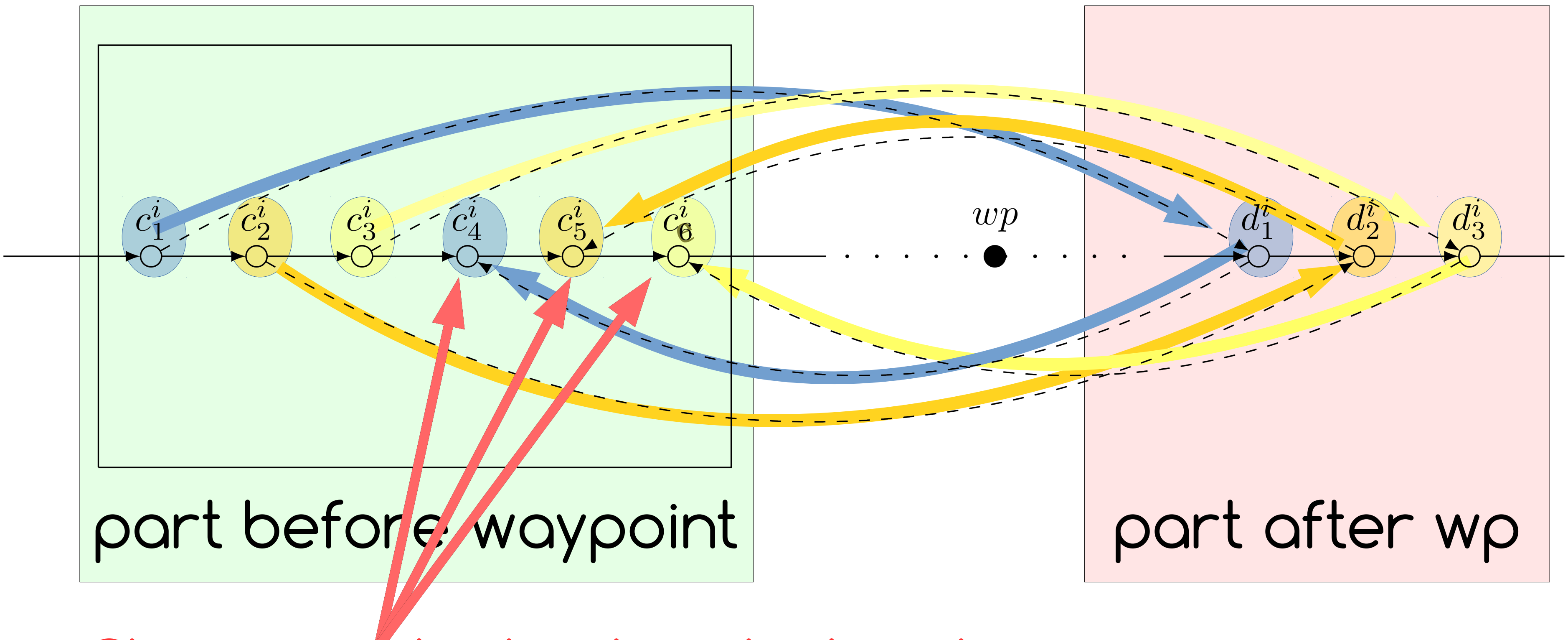
Construction of 3-SAT Reduction: Clause Gadgets



Construction of 3-SAT Reduction: Clause Gadgets

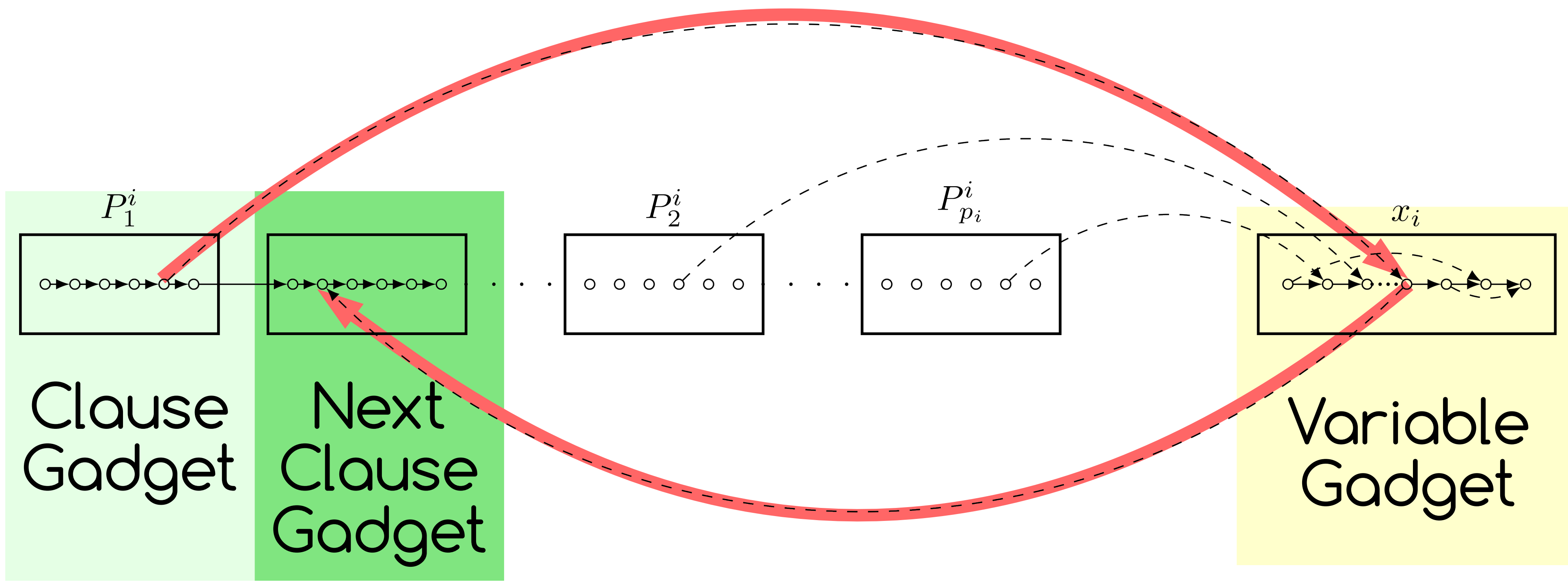


Construction of 3-SAT Reduction: Clause Gadgets

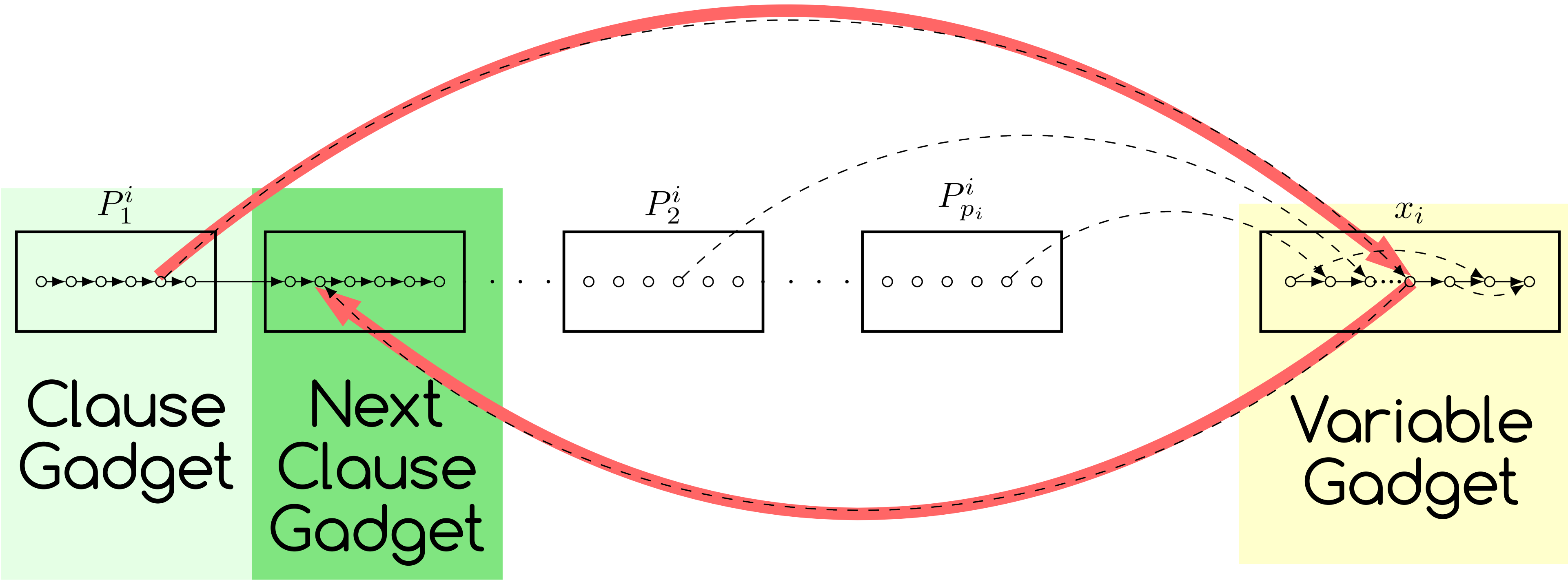


Clause gadget is tangled, as long as neither of these nodes is updated.

Construction of 3-SAT Reduction: Connection Clause with Variable Gadgets

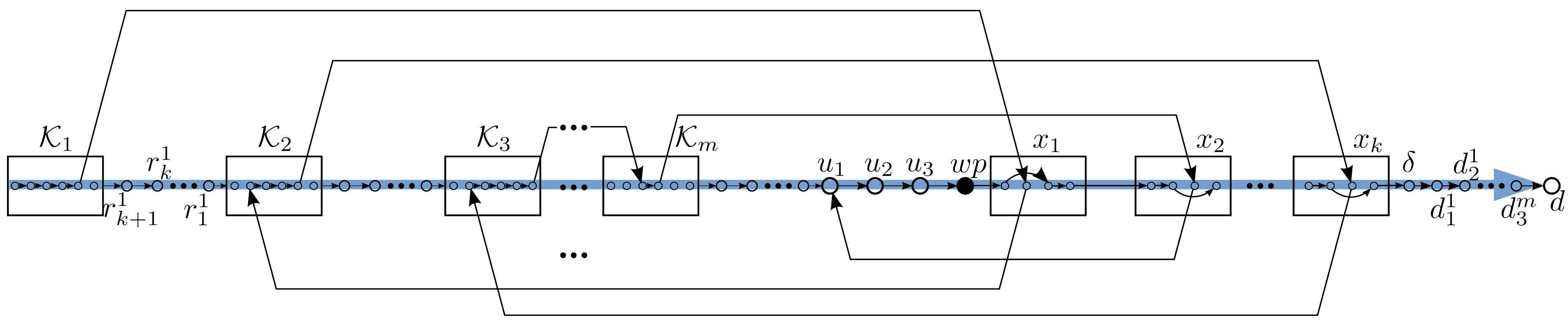


Construction of 3-SAT Reduction: Connection Clause with Variable Gadgets

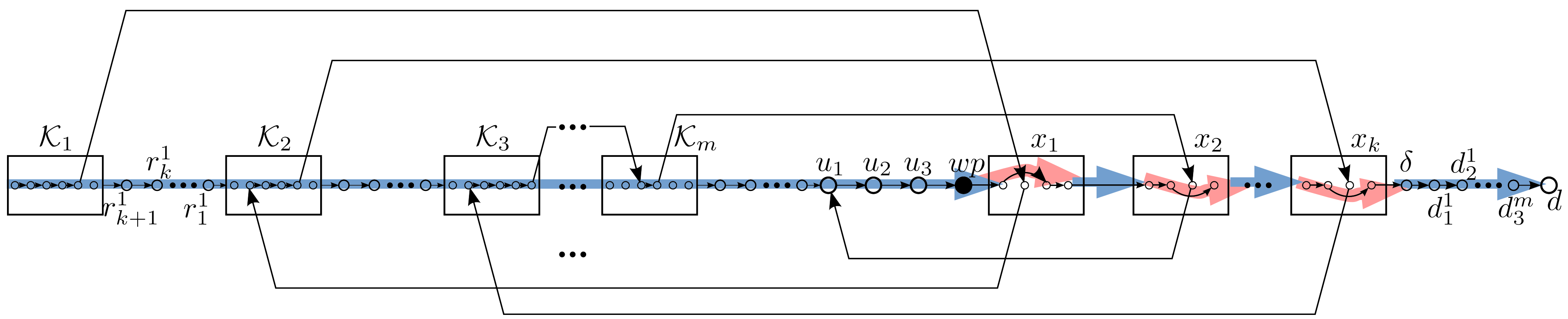


To untangle clauses, a consistent assignment
Of truth values to variables must be found.

Construction of 3-SAT Reduction: Untangling Clauses

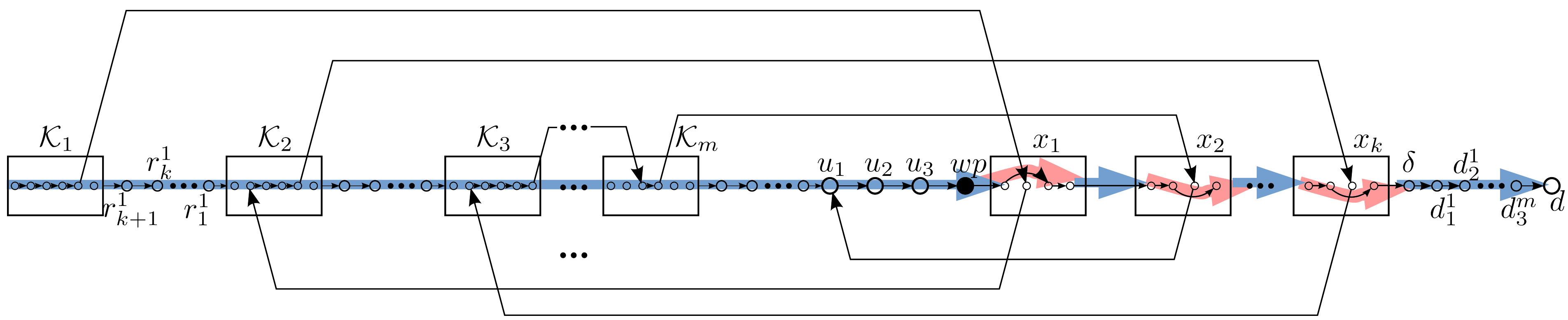


Construction of 3-SAT Reduction: Untangling Clauses

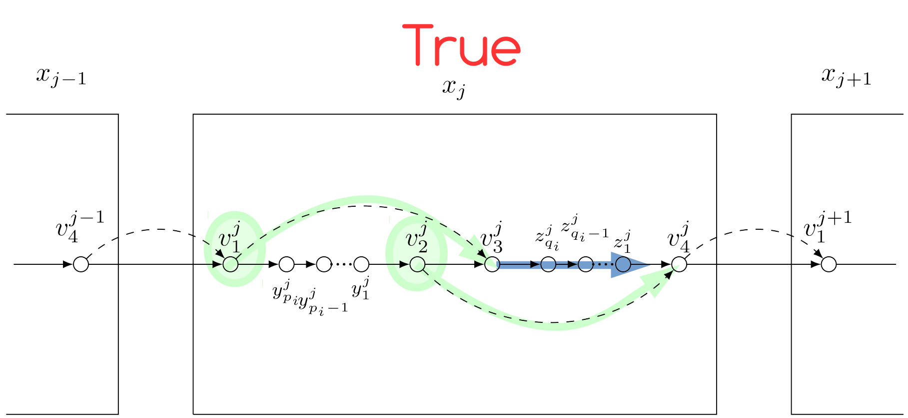


1) Trigger updates in variable gadgets depending on truth value of the variable

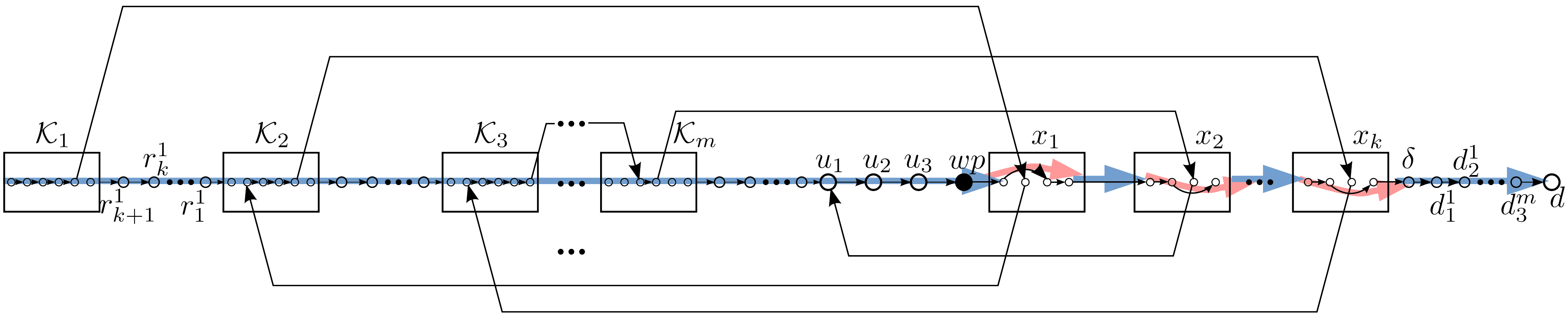
Construction of 3-SAT Reduction: Untangling Clauses



1) Trigger updates in variable gadgets depending on truth value of the variable

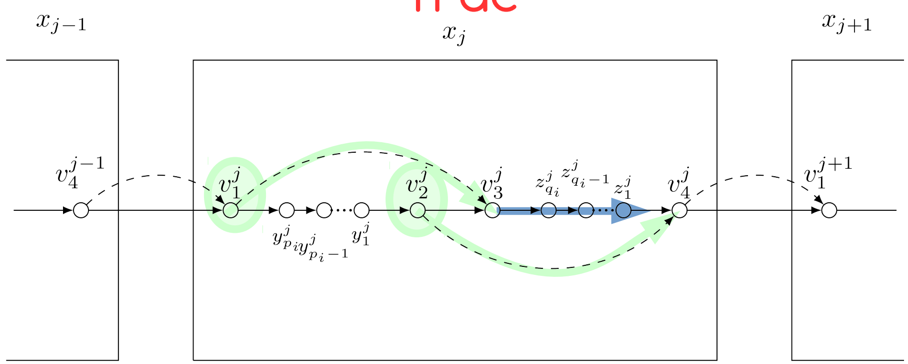


Construction of 3-SAT Reduction: Untangling Clauses

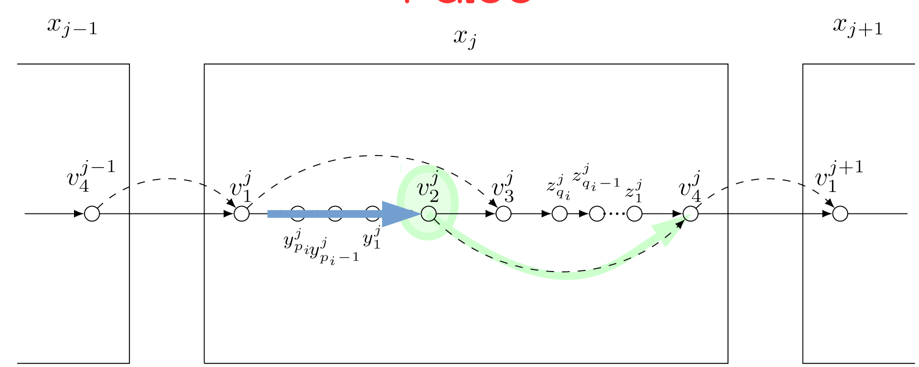


1) Trigger updates in variable gadgets depending on truth value of the variable

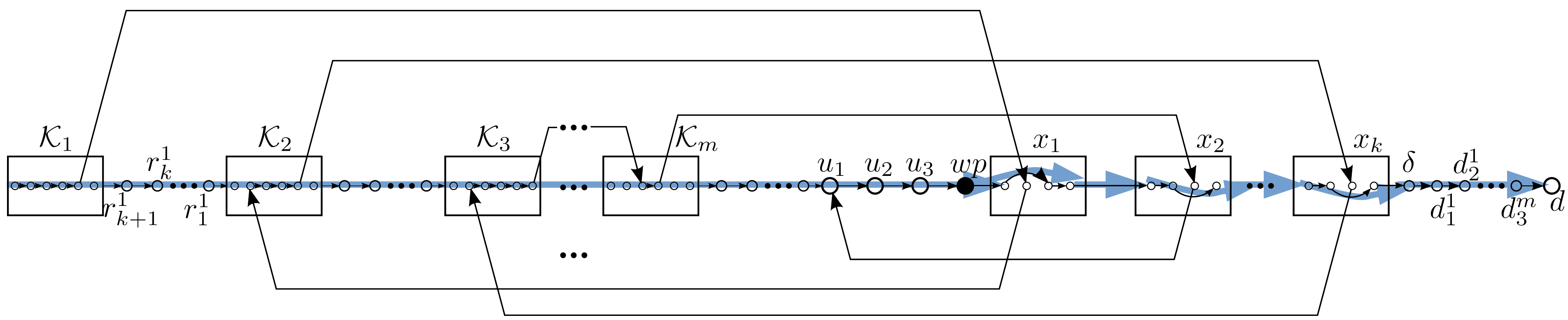
True



False

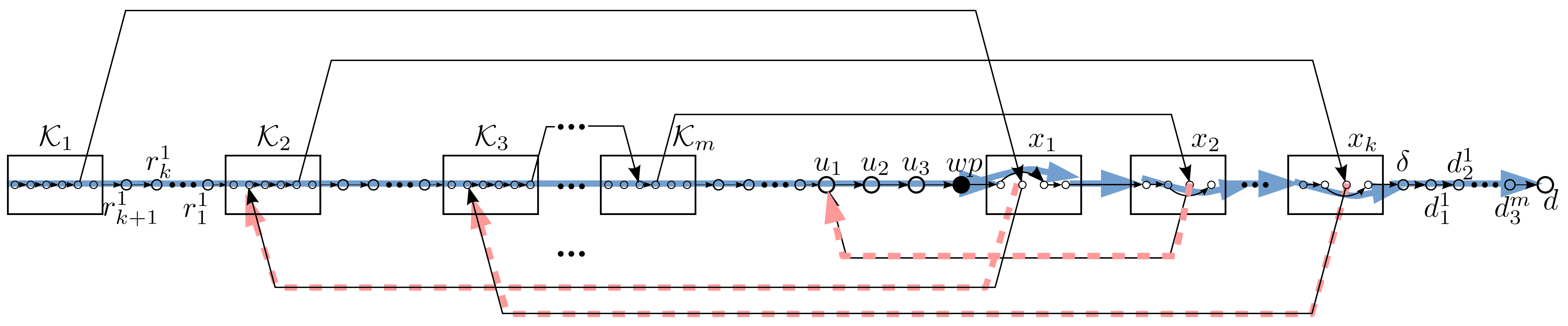


Construction of 3-SAT Reduction: Untangling Clauses



1) Trigger updates in variable gadgets depending on truth value of the variable

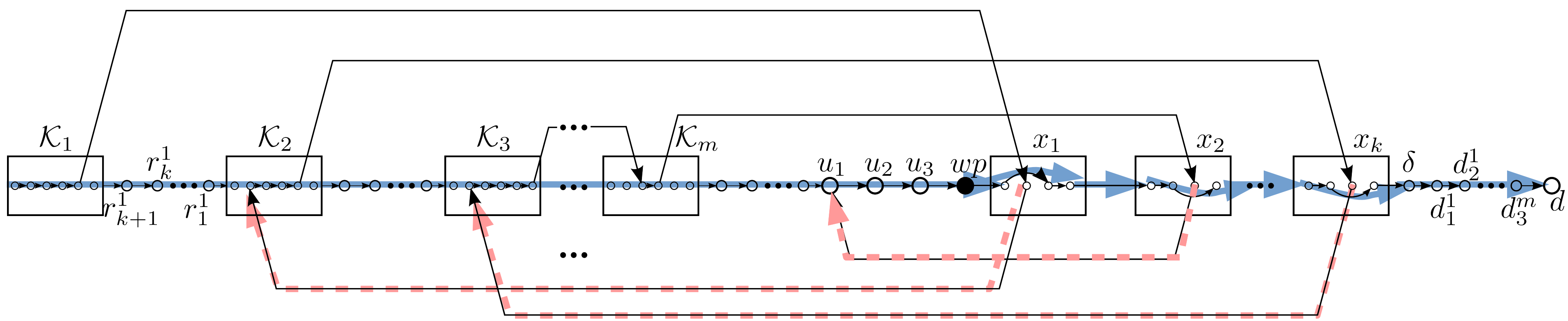
Construction of 3-SAT Reduction: Untangling Clauses



1) Trigger updates in variable gadgets depending on truth value of the variable

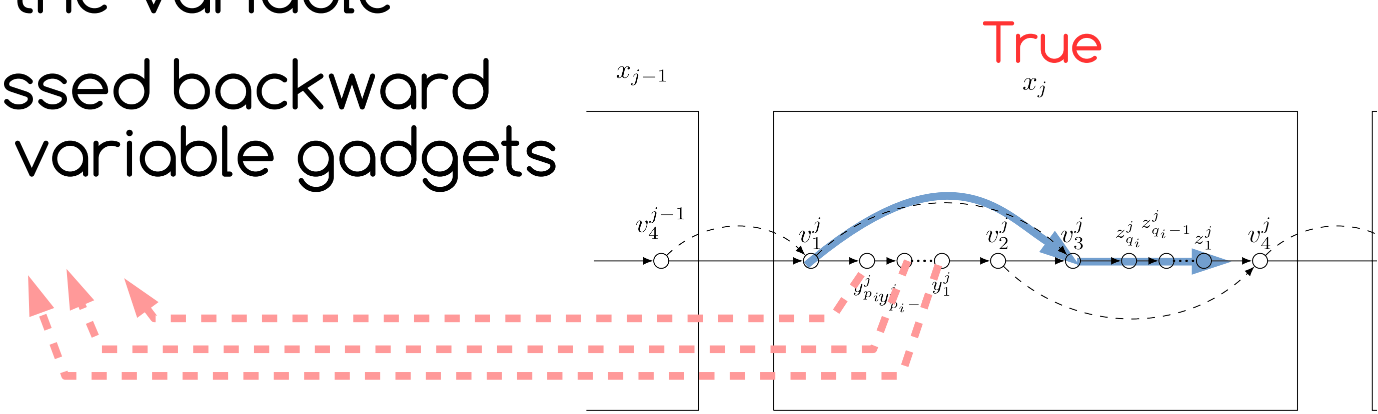
2) Enable now bypassed backward rules from within variable gadgets

Construction of 3-SAT Reduction: Untangling Clauses

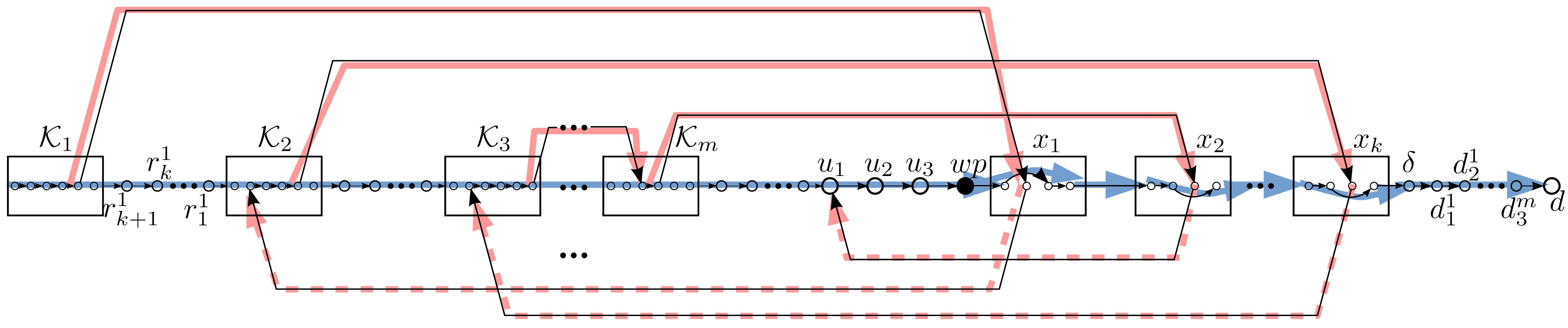


1) Trigger updates in variable gadgets depending on truth value of the variable

2) Enable now bypassed backward rules from within variable gadgets

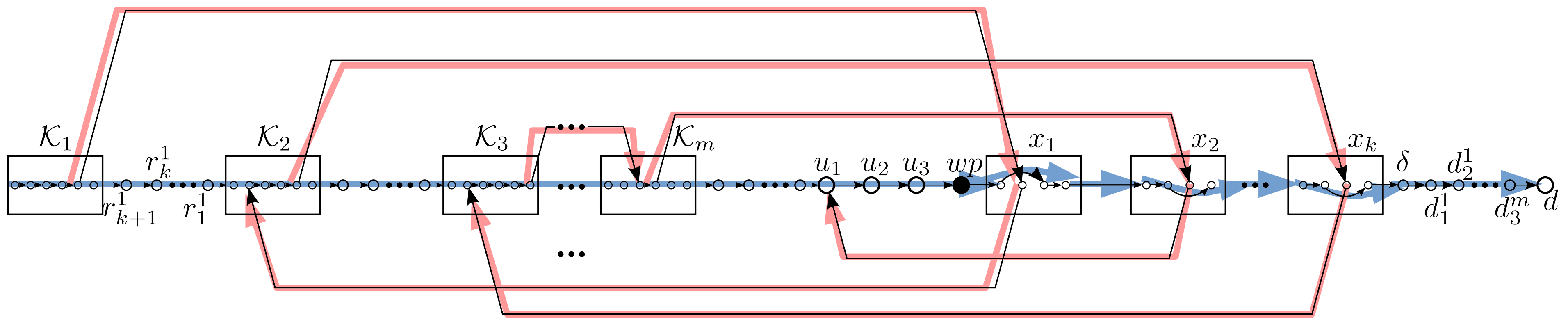


Construction of 3-SAT Reduction: Untangling Clauses



1) Trigger updates in variable gadgets depending on truth value of the variable
2) Enable now bypassed backward rules from within variable gadgets

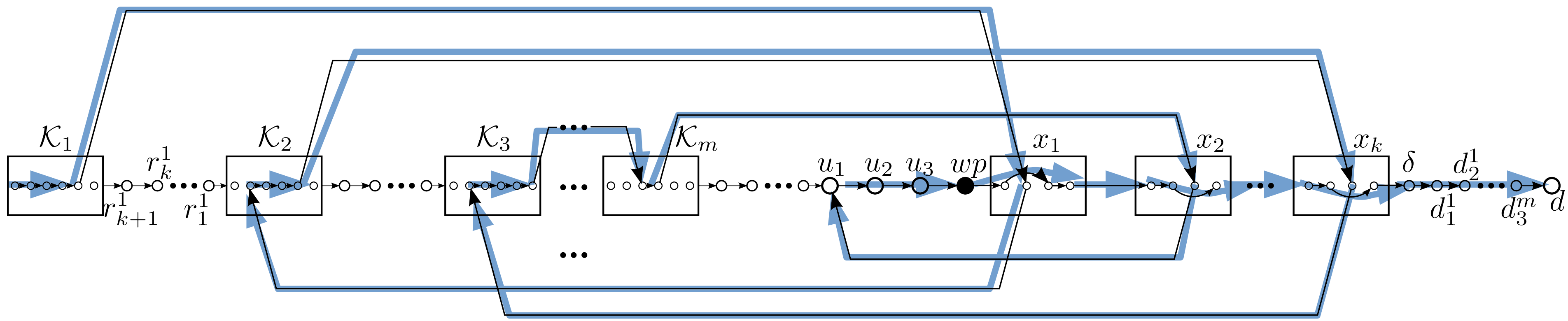
Construction of 3-SAT Reduction: Untangling Clauses



1) Trigger updates in variable gadgets depending on truth value of the variable

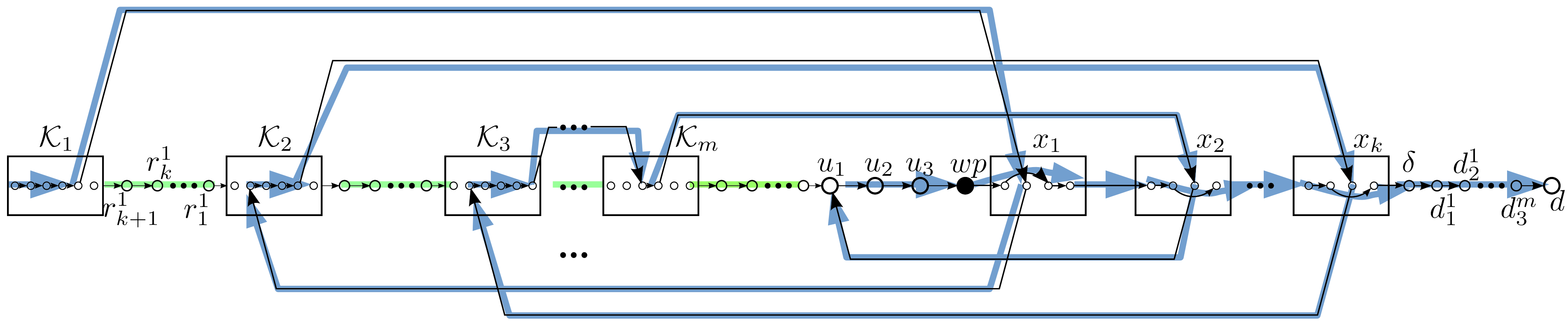
2) Enable now bypassed backward rules from within variable gadgets

Construction of 3-SAT Reduction: Untangling Clauses



1) Trigger updates in variable gadgets depending on truth value of the variable
2) Enable now bypassed backward rules from within variable gadgets
3) For each clause select (arbitrarily) one of the valid assignments

Construction of 3-SAT Reduction: Untangling Clauses



1) Trigger updates in variable gadgets depending on truth value of the variable
2) Enable now bypassed backward rules from within variable gadgets
3) For each clause select (arbitrarily) one of the valid assignments. This untangles all clauses.
4) (start updating remaining nodes)

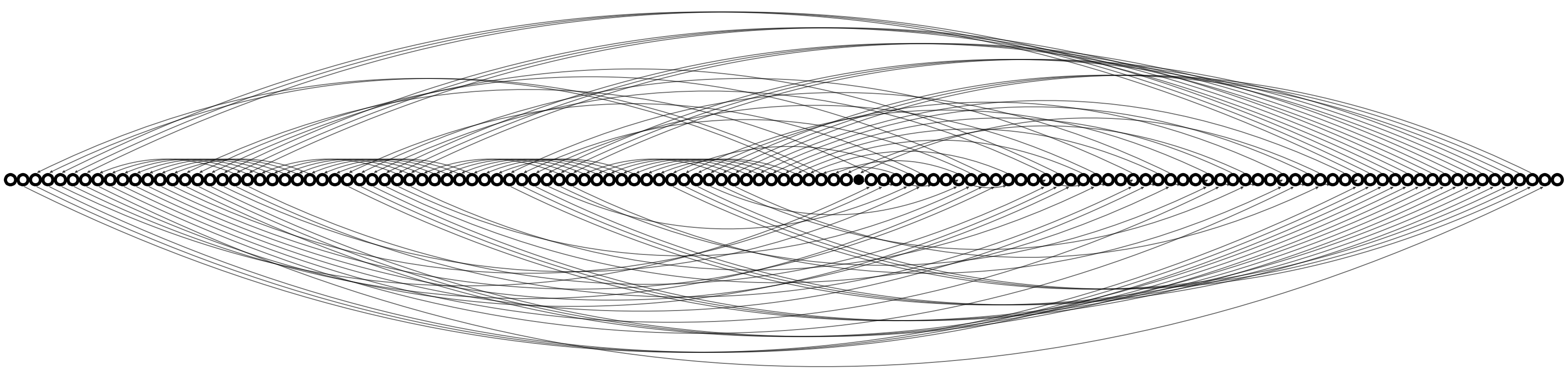
Main Result

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_3) \wedge (\neg x_4 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_5 \vee x_6) \wedge (x_2 \vee \neg x_5 \vee \neg x_6)$$

3-SAT formula is satisfiable

iff.

constructed network update instance is updateable



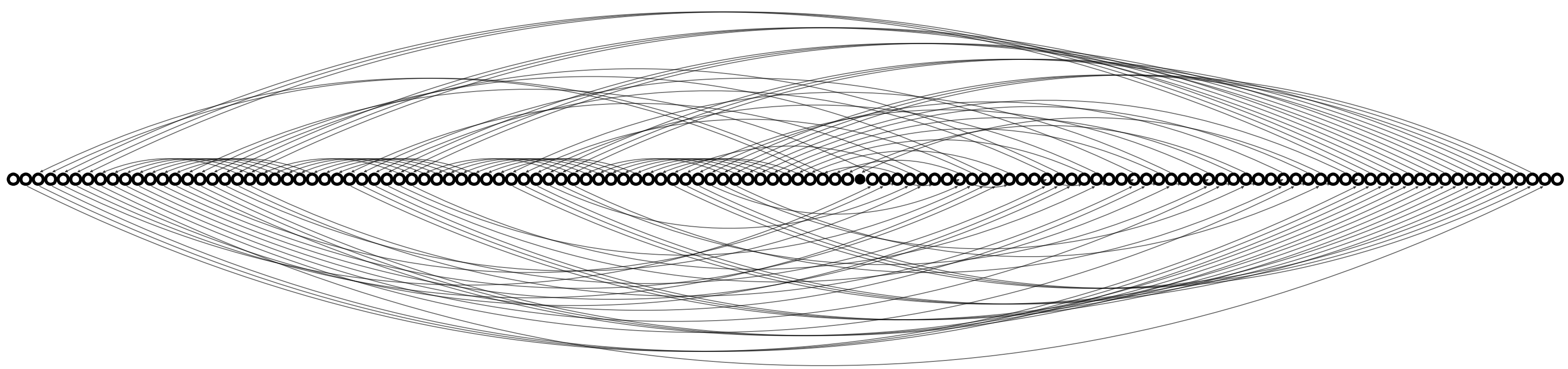
Main Result

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_3) \wedge (\neg x_4 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_5 \vee x_6) \wedge (x_2 \vee \neg x_5 \vee \neg x_6)$$

3-SAT formula is satisfiable

iff.

constructed network update instance is updateable

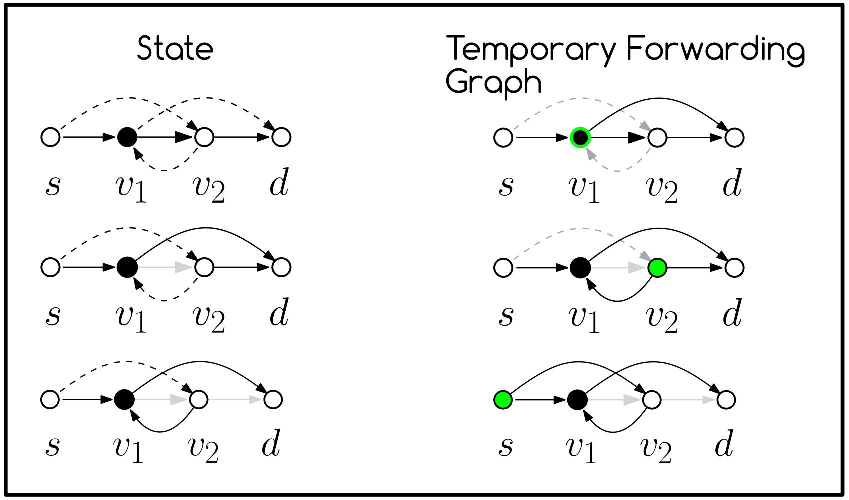


Independent of whether RLF or SLF is considered.

Practice:
Computing Update Schedules

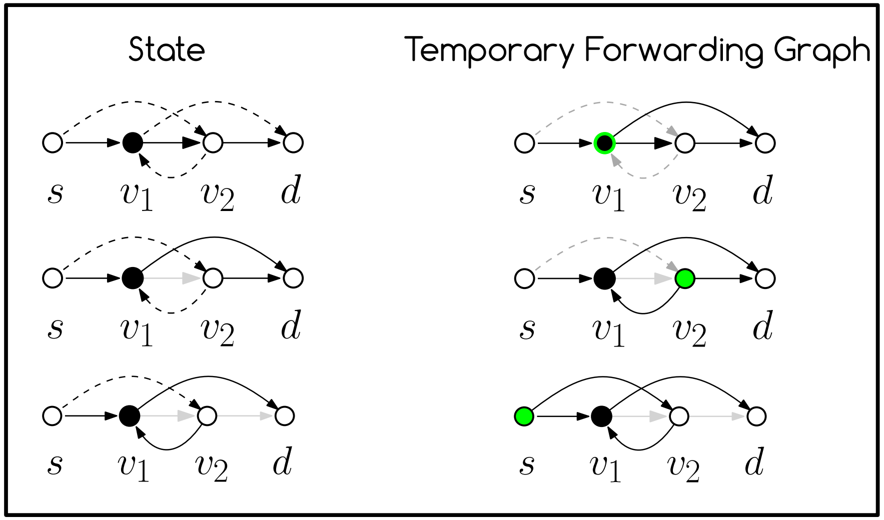
Computing Update Schedules

- Finding a solution is NP-hard
- We employ Mixed-Integer Programming to compute solutions
  - evaluate computational hardness
  - quantitatively analyze feasibility



Computing Update Schedules

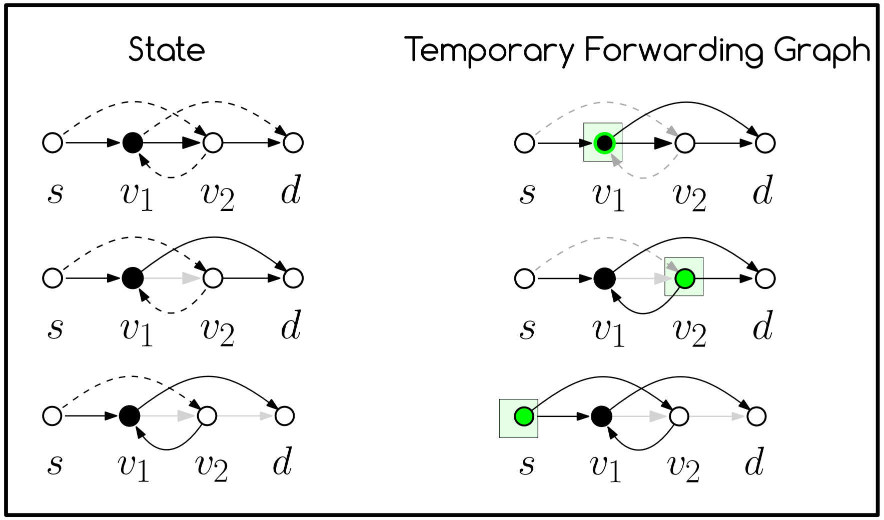
- LF and WPE are checked using Temporary Forwarding Graph
- Given decisions which switches to update, the state and the Temporary Forwarding Graph follow



Computing Update Schedules

Assign update of switch v to a single round r:

$$x_v^r \in \{0,1\}$$



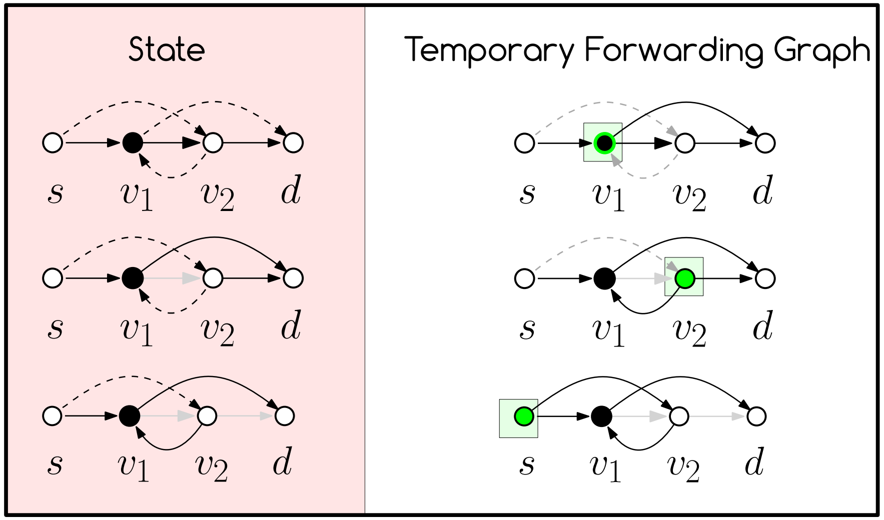
Computing Update Schedules

Assign update of switch v to a single round r:

$$x_v^r \in \{0,1\}$$

Represent forwarding state after round r by

$$y_{u,v}^r \in [0,1]$$



Computing Update Schedules

Assign update of switch v to a single round r:

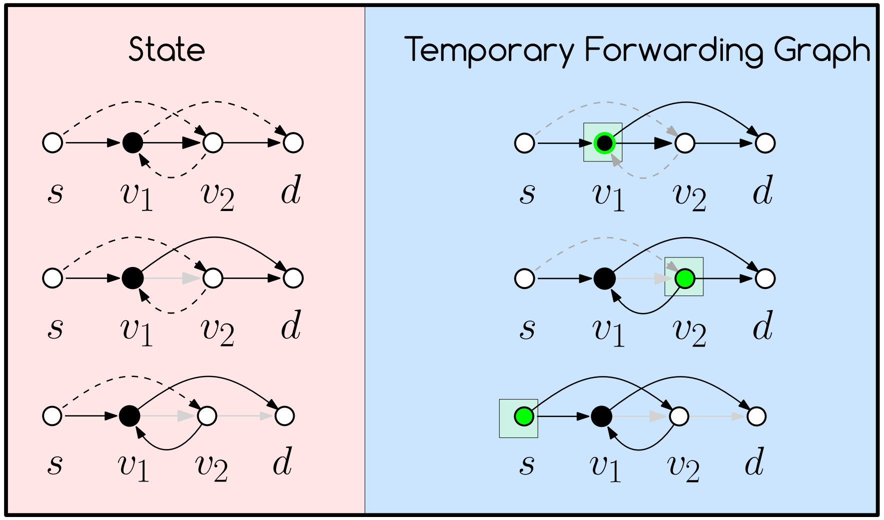
$$x_v^r \in \{0,1\}$$

Represent forwarding state after round r by

$$y_{u,v}^r \in [0,1]$$

Represent Temporary Forwarding Graph by

$$y_{u,v}^{r-1 \vee r} \in [0,1]$$



Computing Update Schedules

Assign update of switch v to a single round r:

$$x_v^r \in \{0,1\}$$

Represent forwarding state after round r by

$$y_{u,v}^r \in [0,1]$$

Represent Temporary Forwarding Graph by

$$y_{u,v}^{r-1 \vee r} \in [0,1]$$

$$1 = \sum_{r \in \mathcal{R}} x_v^r$$

$$y_{u,v}^r = 1 - \sum_{r' \leq r} x_u^{r'}$$ (old edges)

$$y_{u,v}^r = \sum_{r' \leq r} x_u^{r'}$$ (new edges)

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r-1}$$

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r}$$

$$y_{u,v}^{r-1 \vee r} \leq \frac{l_v^r - l_u^r - 1}{|V| - 1} + 1$$

$$\overline{a}_s^{r,w} = 1$$

$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r-1} - 1$$

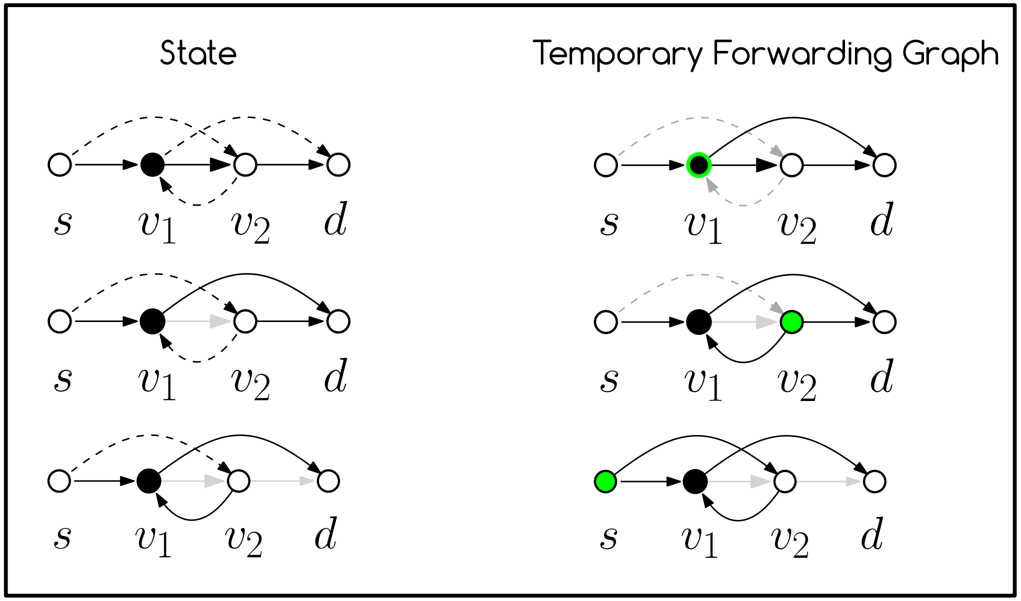
$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r} - 1$$

$$\overline{a}_d^{r,w} = 0$$

Computing Update Schedules

Enforce SLF by employing Miller-Tucker-Zemlin Constraints by level variables:

$$l_v^r \in [0, |V|-1]$$



$$1 = \sum_{r \in \mathcal{R}} x_v^r$$

$$y_{u,v}^r = 1 - \sum_{r' \leq r} x_u^{r'}$$ (old edges)

$$y_{u,v}^r = \sum_{r' \leq r} x_u^{r'}$$ (new edges)

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r-1}$$

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r}$$

$$y_{u,v}^{r-1 \vee r} \leq \frac{l_v^r - l_u^r - 1}{|V| - 1} + 1$$

$$\overline{a}_s^{r,w} = 1$$

$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r-1} - 1$$

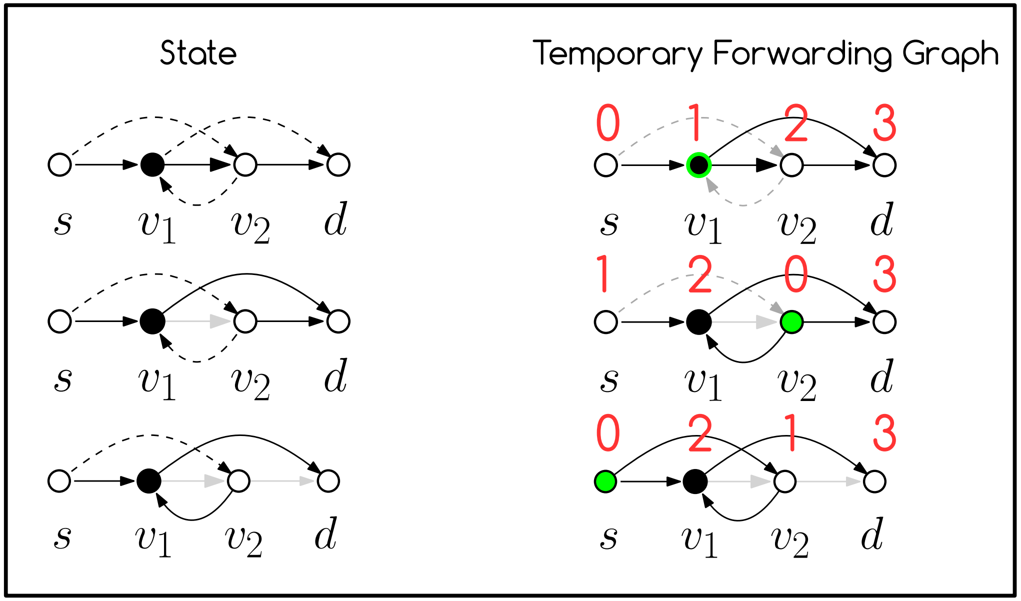
$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r} - 1$$

$$\overline{a}_d^{r,w} = 0$$

Computing Update Schedules

Enforce SLF by employing Miller-Tucker-Zemlin Constraints by level variables:

$$l_v^r \in [0, |V|-1]$$



$$1 = \sum_{r \in \mathcal{R}} x_v^r$$

$$y_{u,v}^r = 1 - \sum_{r' \leq r} x_u^{r'}$$ (old edges)

$$y_{u,v}^r = \sum_{r' \leq r} x_u^{r'}$$ (new edges)

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r-1}$$

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r}$$

$$y_{u,v}^{r-1 \vee r} \leq \frac{l_v^r - l_u^r - 1}{|V| - 1} + 1$$

$$\overline{a}_s^{r,w} = 1$$

$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r-1} - 1$$

$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r} - 1$$

$$\overline{a}_d^{r,w} = 0$$

Computing Update Schedules

Enforce SLF by employing Miller-Tucker-Zemlin Constraints by level variables:

$$l_v^r \in [0, |V|-1]$$

Guarantee WPE by reachability constraints:

Nodes reachable from the source, without using waypoint w, are 'marked'

by $\overline{a}_v^{r,w} = 1$

$$1 = \sum_{r \in \mathcal{R}} x_v^r$$

$$y_{u,v}^r = 1 - \sum_{r' \leq r} x_u^{r'}$$ (old edges)

$$y_{u,v}^r = \sum_{r' \leq r} x_u^{r'}$$ (new edges)

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r-1}$$

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r}$$

$$y_{u,v}^{r-1 \vee r} \leq \frac{l_v^r - l_u^r - 1}{|V| - 1} + 1$$

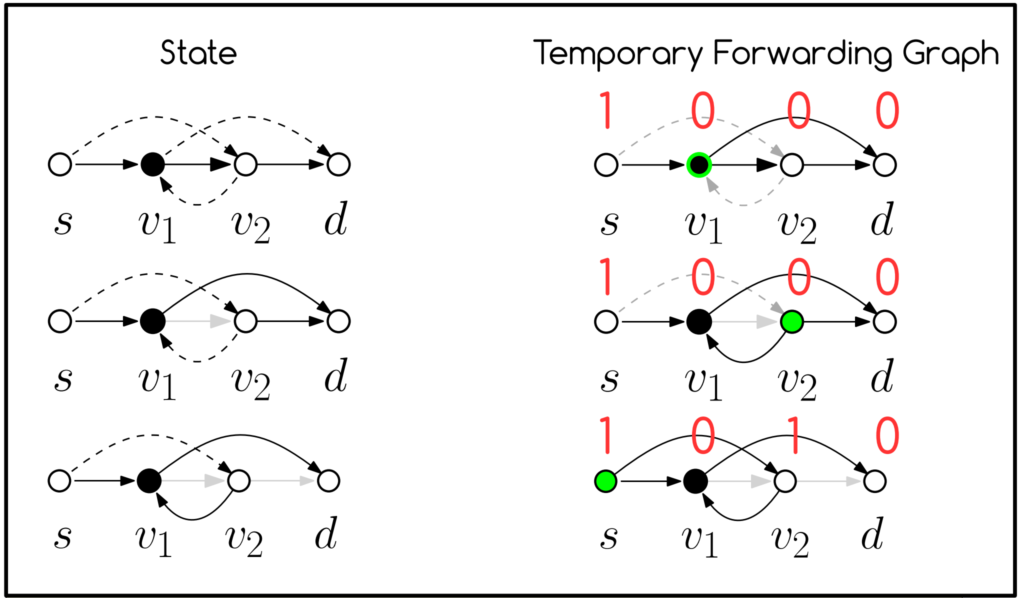
$$\overline{a}_s^{r,w} = 1$$

$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r-1} - 1$$ (edges not incident to w)

$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r} - 1$$ (edges not incident to w)

$$\overline{a}_d^{r,w} = 0$$

Computing Update Schedules



Guarantee WPE by reachability constraints:

Nodes reachable from the source, without using waypoint w, are 'marked'

by $\overline{a}_v^{r,w}=1$

$$1 = \sum_{r \in \mathcal{R}} x_v^r$$

$$y_{u,v}^r = 1 - \sum_{r' \leq r} x_u^{r'}$$ (old edges)

$$y_{u,v}^r = \sum_{r' \leq r} x_u^{r'}$$ (new edges)

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r-1}$$

$$y_{u,v}^{r-1 \vee r} \geq y_{u,v}^{r}$$

$$y_{u,v}^{r-1 \vee r} \leq \frac{l_v^r - l_u^r - 1}{|V| - 1} + 1$$

$$\overline{a}_s^{r,w} = 1$$

$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r-1} - 1$$ (edges not incident to w)

$$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r} - 1$$ (edges not incident to w)

$$\overline{a}_d^{r,w} = 0$$

Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.

Mixed-Integer Program 1: Optimal Rounds (-R-)

$\min R$		(Obj)
$R \geq r \cdot x_v^r$	$r \in \mathcal{R}, v \in V$	(1)
$1 = \sum_{r \in \mathcal{R}} x_v^r$	$v \in V$	(2)
$y_{u,v}^r = 1 - \sum_{r' \leq r} x_u^{r'}$	$r \in \mathcal{R},\ (u,v) \in E_{\pi_1}$	(3)
$y_{u,v}^r = \sum_{r' \leq r} x_u^{r'}$	$r \in \mathcal{R},\ (u,v) \in E_{\pi_2}$	(4)
$a_s^r = 1$	$r \in \mathcal{R}$	(5)
$a_v^r \geq a_u^r + y_{u,v}^{r-1} - 1$	$r \in \mathcal{R},\ (u,v) \in E$	(6)
$a_v^r \geq a_u^r + y_{u,v}^r - 1$	$r \in \mathcal{R},\ (u,v) \in E$	(7)
$y_{u,v}^{r-1 \vee r} \geq a_u^r + y_{u,v}^{r-1} - 1$	$r \in \mathcal{R},\ (u,v) \in E$	(8)
$y_{u,v}^{r-1 \vee r} \geq a_u^r + y_{u,v}^r - 1$	$r \in \mathcal{R},\ (u,v) \in E$	(9)
$y_{u,v}^{r-1 \vee r} \leq \frac{l_v^r - l_u^r - 1}{\lvert V \rvert - 1} + 1$	$r \in \mathcal{R},\ (u,v) \in E$	(10)
$\overline{a}_s^{r,w} = 1$	$r \in \mathcal{R}, w \in WP$	(11)
$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r-1} - 1$	$r \in \mathcal{R}, w \in WP,$ $(u,v) \in E_{\mathsf{WP}}^w$	(12)
$\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^r - 1$	$r \in \mathcal{R}, w \in WP,$ $(u,v) \in E_{\mathsf{WP}}^w$	(13)
$\overline{a}_d^{r,w} = 0$	$r \in \mathcal{R}, w \in WP$	(14)

Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.
- Objective: minimize #rounds

Mixed-Integer Program 1: Optimal Rounds (-R-)

$$
\begin{array}{rlr}
\min R & & \text{(Obj)} \\
R \geq r \cdot x_v^r & r \in \mathcal{R}, v \in V & (1) \\
1 = \sum_{r \in \mathcal{R}} x_v^r & v \in V & (2) \\
y_{u,v}^r = 1 - \sum_{r' \leq r} x_u^{r'} & r \in \mathcal{R},\ (u,v) \in E_{\pi_1} & (3) \\
y_{u,v}^r = \sum_{r' \leq r} x_u^{r'} & r \in \mathcal{R},\ (u,v) \in E_{\pi_2} & (4) \\
a_s^r = 1 & r \in \mathcal{R} & (5) \\
a_v^r \geq a_u^r + y_{u,v}^{r-1} - 1 & r \in \mathcal{R},\ (u,v) \in E & (6) \\
a_v^r \geq a_u^r + y_{u,v}^r - 1 & r \in \mathcal{R},\ (u,v) \in E & (7) \\
y_{u,v}^{r-1 \vee r} \geq a_u^r + y_{u,v}^{r-1} - 1 & r \in \mathcal{R},\ (u,v) \in E & (8) \\
y_{u,v}^{r-1 \vee r} \geq a_u^r + y_{u,v}^r - 1 & r \in \mathcal{R},\ (u,v) \in E & (9) \\
y_{u,v}^{r-1 \vee r} \leq \dfrac{l_v^r - l_u^r - 1}{|V| - 1} + 1 & r \in \mathcal{R},\ (u,v) \in E & (10) \\
\overline{a}_s^{r,w} = 1 & r \in \mathcal{R}, w \in WP & (11) \\
\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^{r-1} - 1 & r \in \mathcal{R}, w \in WP,\ (u,v) \in E_{\mathsf{WP}}^w & (12) \\
\overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{u,v}^r - 1 & r \in \mathcal{R}, w \in WP,\ (u,v) \in E_{\mathsf{WP}}^w & (13) \\
\overline{a}_d^{r,w} = 0 & r \in \mathcal{R}, w \in WP & (14)
\end{array}
$$

Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.
- Objective: minimize #rounds
- Some employed constraints are 'weak'; we propose:
  - Decision Variant (D)
  - A Flow Extension (F)

Mixed-Integer Program 1: Optimal Rounds (-R-)

$\min R$		(Obj)
$R \geq r \cdot x_v^r$	$r \in \mathcal{R}, v \in V$	(1)
$1 = \sum_{r \in \mathcal{R}} x_v^r$	$v \in V$	(2)
$y_{u,v}^r = 1 - \sum_{r' \leq r} x_u^{r'}$	$r \in \mathcal{R},\ (u,v) \in E_{\pi_1}$	(3)
$y_{u,v}^r = \sum_{r' \leq r} x_u^{r'}$	$r \in \mathcal{R},\ (u,v) \in E_{\pi_2}$	(4)
$a_s^r = 1$	$r \in \mathcal{R}$	(5)
$a_v^r \geq a_u^r + y_{u,v}^{r-1} - 1$	$r \in \mathcal{R},\ (u,v) \in E$	(6)
$a_v^r \geq a_u^r + y_{u,v}^r - 1$	$r \in \mathcal{R},\ (u,v) \in E$	(7)
$y_{u,v}^{r-1 \vee r} \geq a_u^r + y_{u,v}^{r-1} - 1$	$r \in \mathcal{R},\ (u,v) \in E$	(8)
$y_{u,v}^{r-1 \vee r} \geq a_u^r + y_{u,v}^r - 1$	$r \in \mathcal{R},\ (u,v) \in E$	(9)
$y_{u,v}^{r-1 \vee r} \leq \dfrac{l_v^r - l_u^r - 1}{\lvert V \rvert - 1} + 1$	$r \in \mathcal{R},\ (u,v) \in E$	(10)
$\bar{a}_s^{r,w} = 1$	$r \in \mathcal{R}, w \in WP$	(11)
$\bar{a}_v^{r,w} \geq \bar{a}_u^{r,w} + y_{u,v}^{r-1} - 1$	$r \in \mathcal{R}, w \in WP,\ (u,v) \in E_{\mathrm{WP}}^w$	(12)
$\bar{a}_v^{r,w} \geq \bar{a}_u^{r,w} + y_{u,v}^r - 1$	$r \in \mathcal{R}, w \in WP,\ (u,v) \in E_{\mathrm{WP}}^w$	(13)
$\bar{a}_d^{r,w} = 0$	$r \in \mathcal{R}, w \in WP$	(14)

Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.
- Objective: minimize #rounds
- Some employed constraints are 'weak'; we propose:
  - Decision Variant (D)
  - A Flow Extension (F)

(D)

Only one update per round.

(F)

Additional s-d flows for each round to improve relaxations.

Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.
- Objective: minimize #rounds
- Some employed constraints are 'weak'; we propose:
  - Decision Variant (D)
  - A Flow Extension (F)

(D)

$$\sum_{v \in V} x_v^r = 1 \quad r \in \mathcal{R}\,.$$

(F)

$$\sum_{e \in \delta^+(s)} f_e^r = 1 \qquad r \in \mathcal{R} \qquad (18)$$

$$\sum_{e \in \delta^+(v)} f_e^r = \sum_{e \in \delta^-(v)} f_e^r \qquad r \in \mathcal{R}, v \in V \setminus \{s, d\} \qquad (19)$$

$$f_e^r \leq y_e^r \qquad r \in \mathcal{R}, e \in E_{\pi_1} \cup E_{\pi_2} \qquad (20)$$

$$\sum_{e \in \delta^-(w)} f_e^r \geq 1 \qquad r \in \mathcal{R}, w \in WP \qquad (21)$$

$$a_v^r \geq f_v^{r-1} \qquad r \in \mathcal{R} \qquad (22^*)$$

$$a_v^r \geq f_v^r \qquad r \in \mathcal{R} \qquad (23^*)$$

Practice: Computational Experiments

Computational Setup

- Generate update instances at random by permuting nodes
- 12,600 instances overall
  - 10 to 30 switches with 1 to 3 waypoints
  - 200 instances for each combination
- (We discard scenarios which can a priori be determined to be infeasible to update, e.g. when waypoints are reordered)

Computational Setup

- Consider 8 different MIP formulations

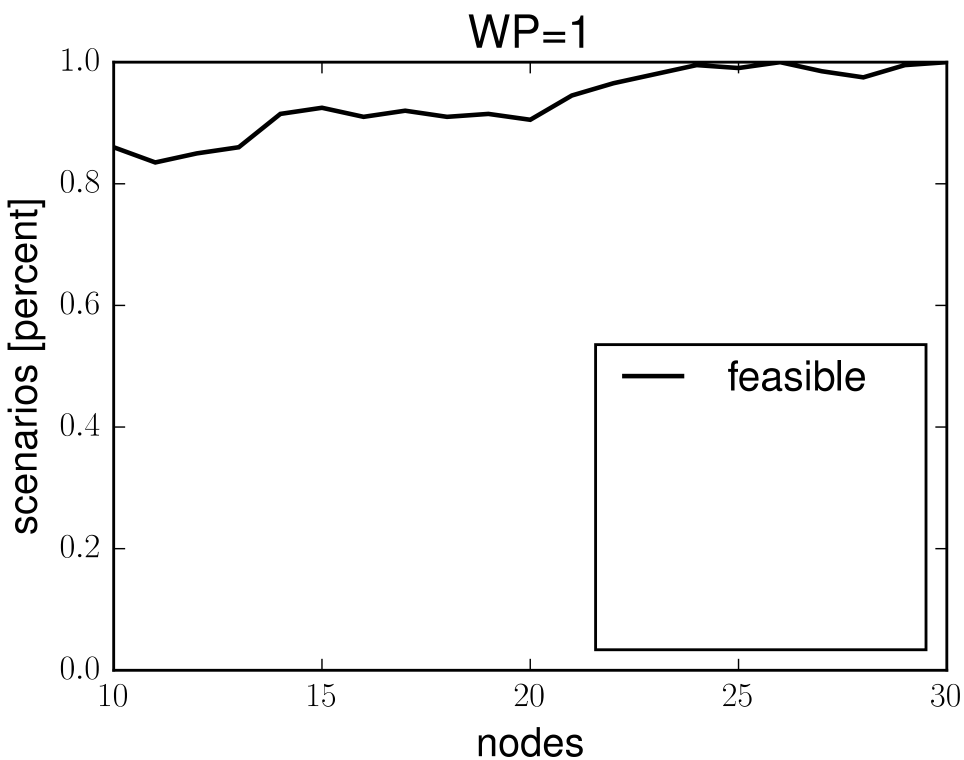
  S(LF) vs. **R**(LF)

  D(ecision) vs. -

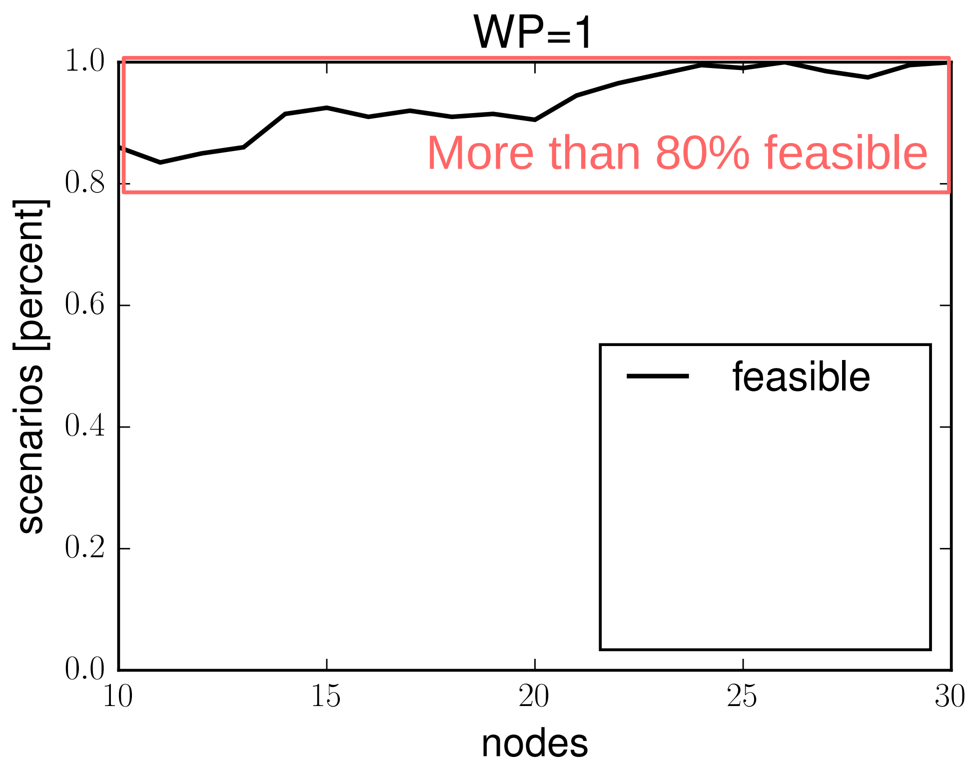
  F(low Extension) vs. -

- Use Gurobi 6.5.0 to solve the formulations using branch-and-bound
- Terminate computations after 600 seconds

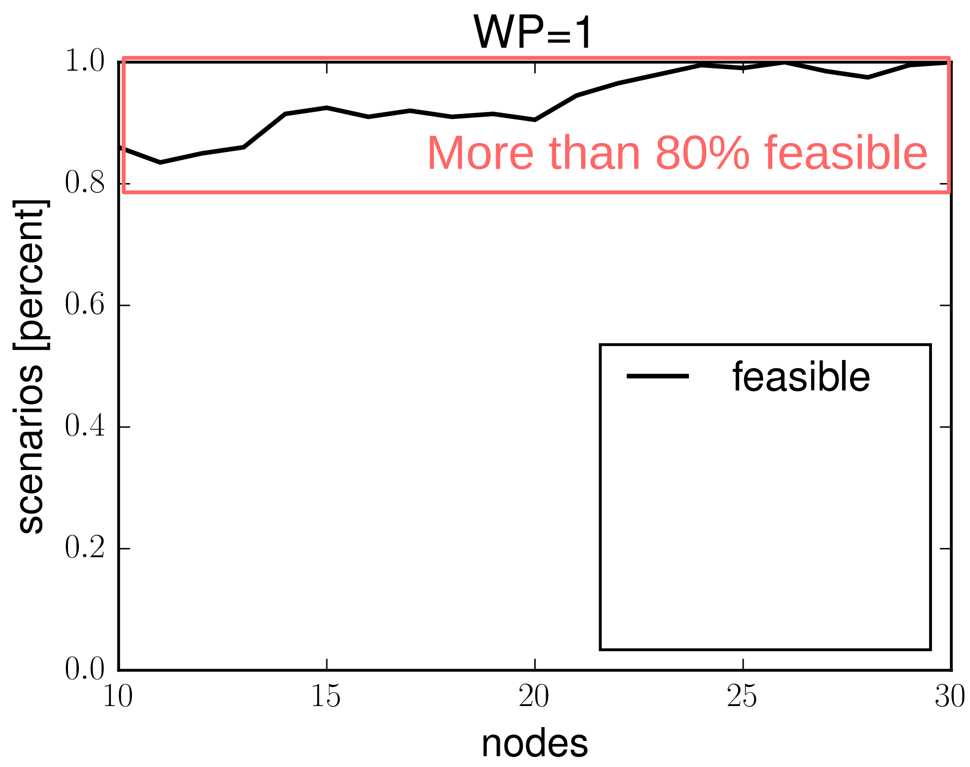
Computational Study: Solvability



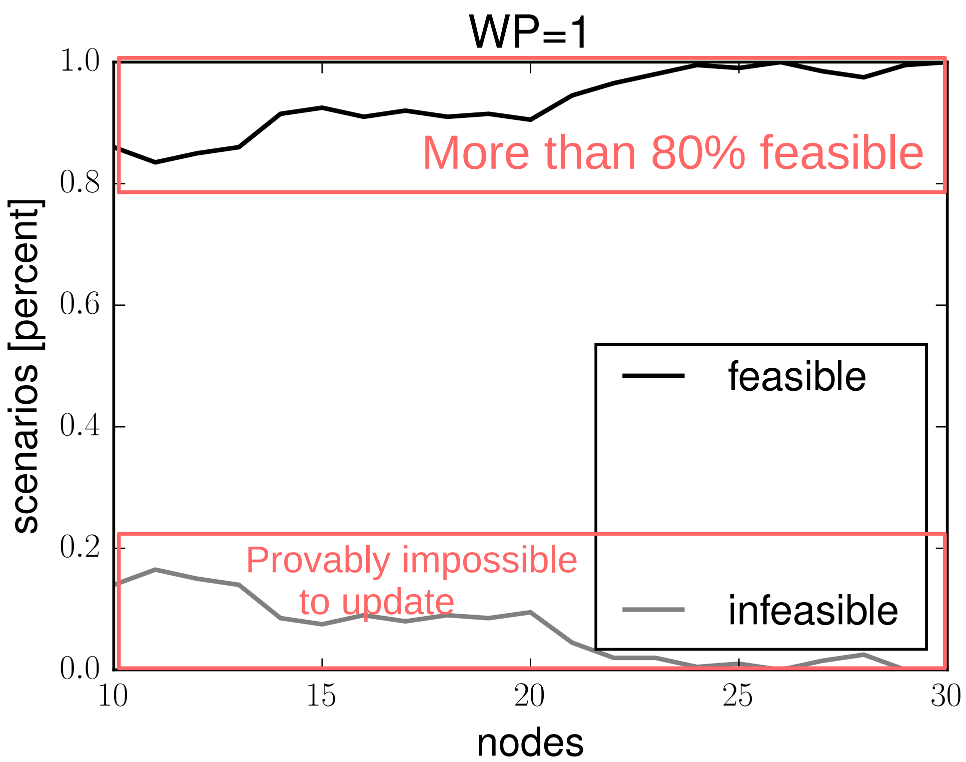
Computational Study: Solvability



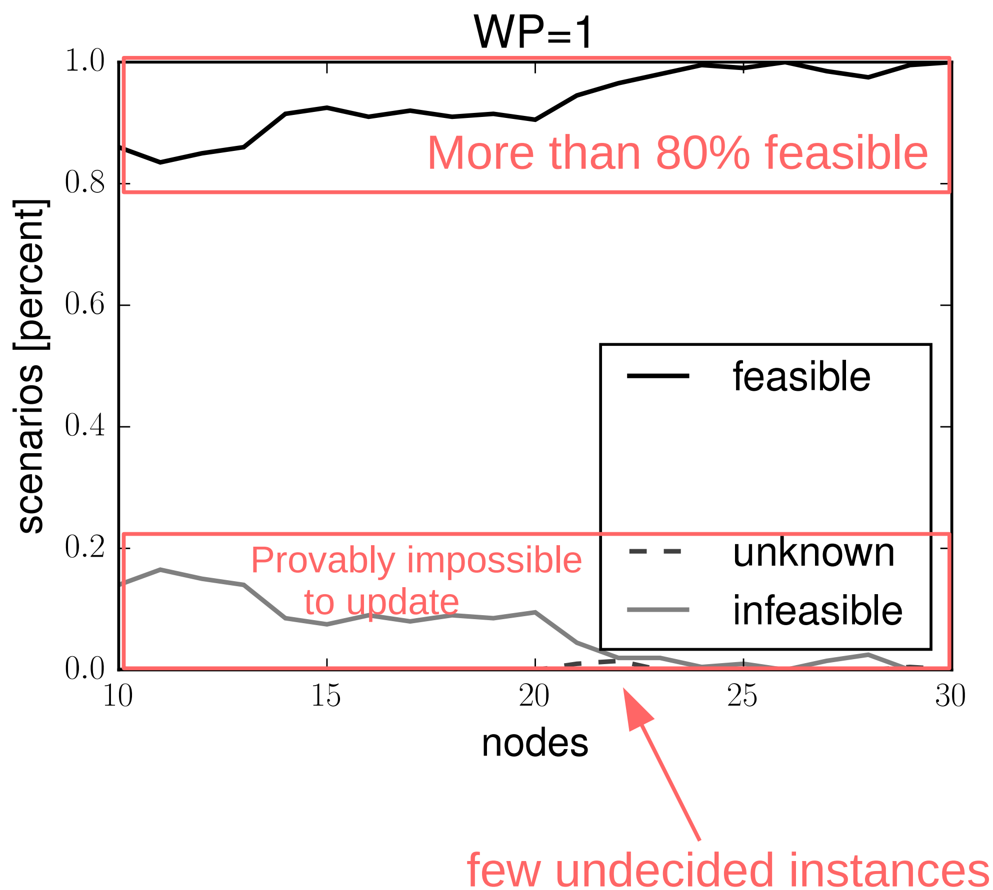
Computational Study: Solvability



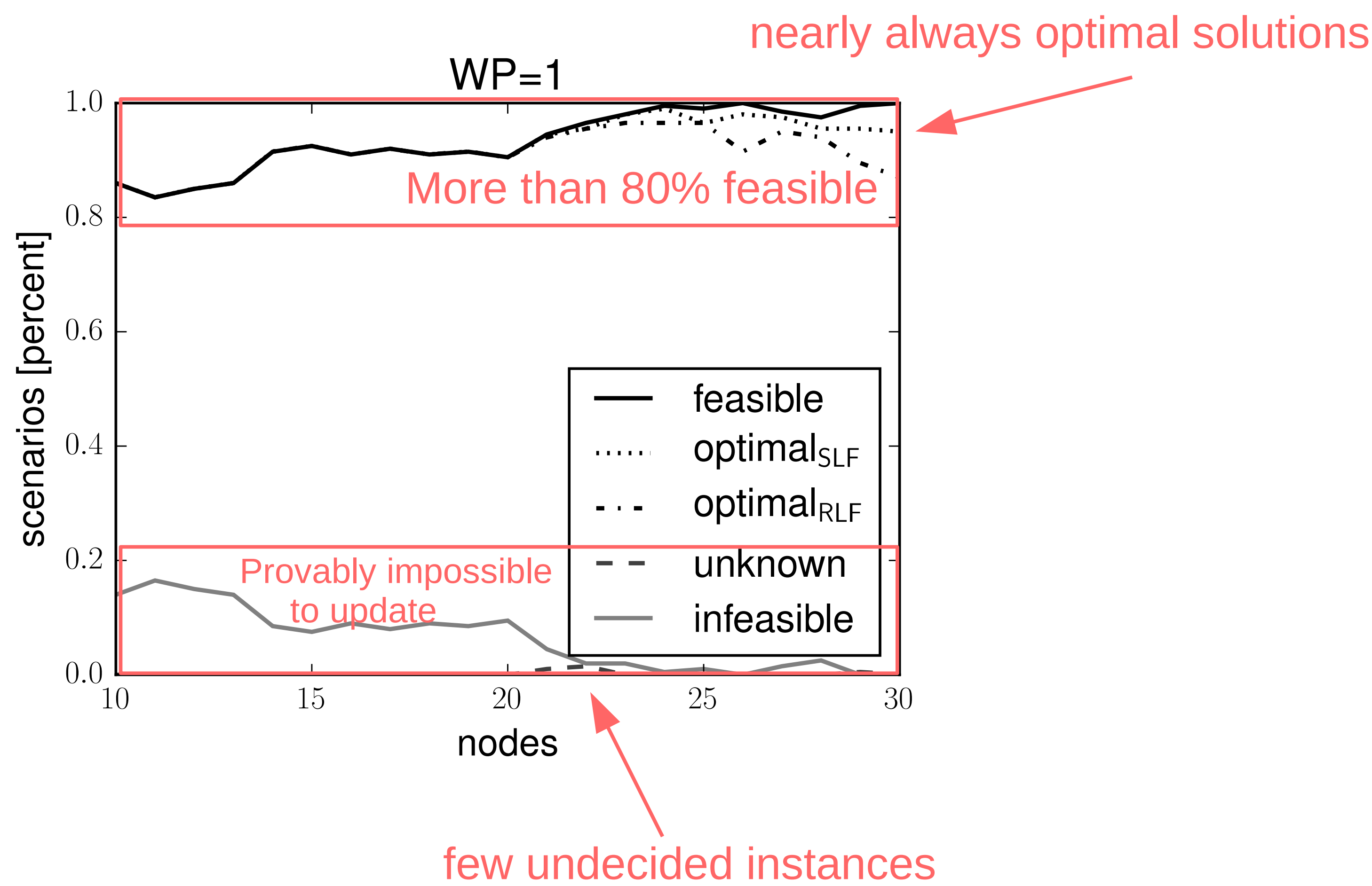
Computational Study: Solvability



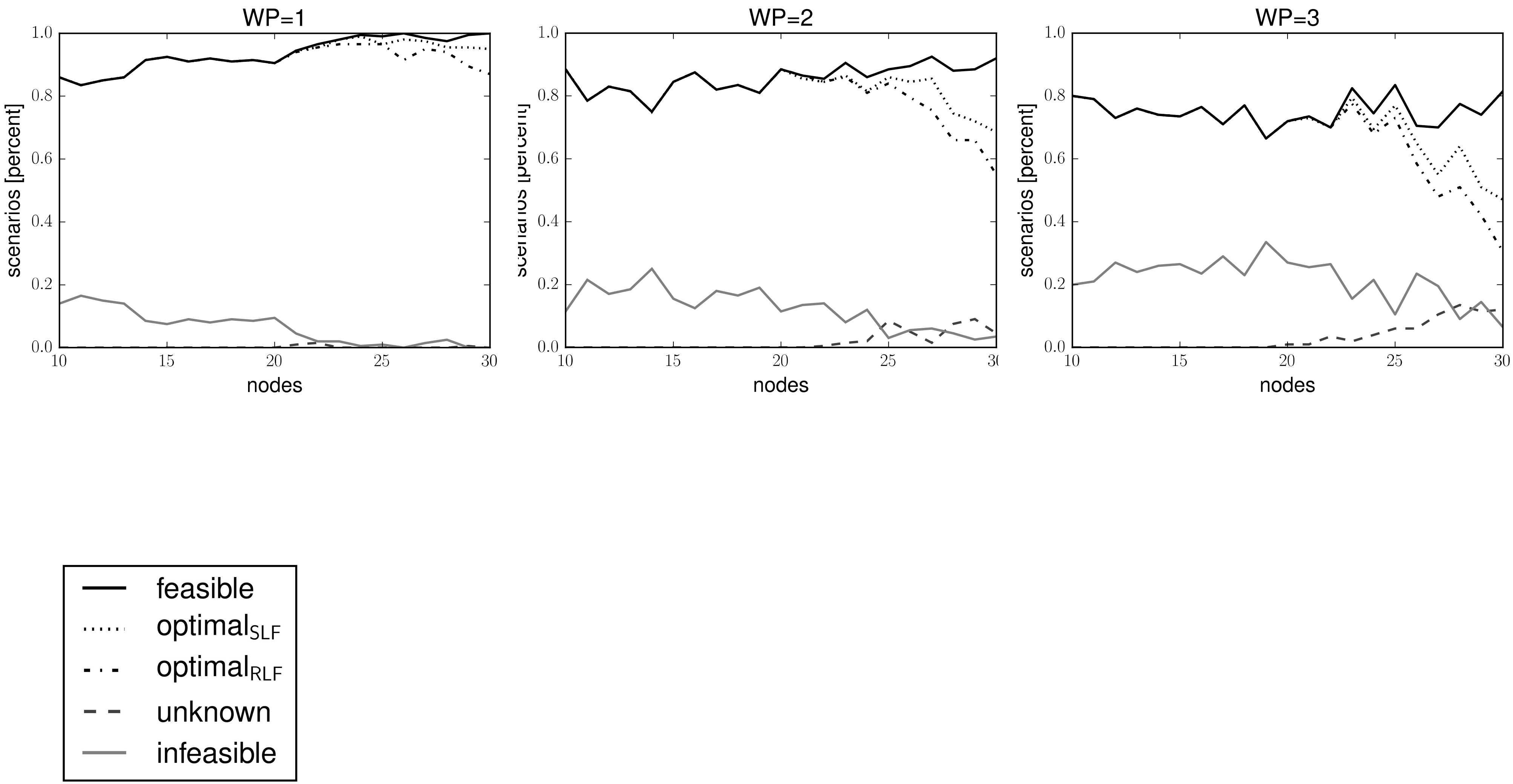
Computational Study: Solvability



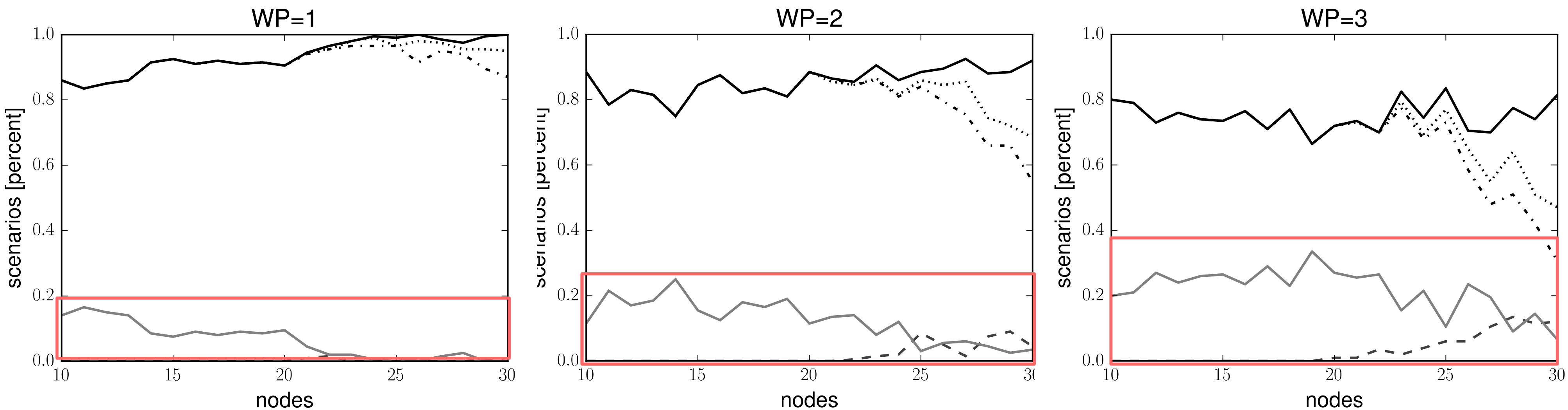
Computational Study: Solvability



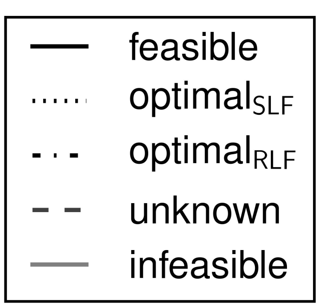
Computational Study: Solvability



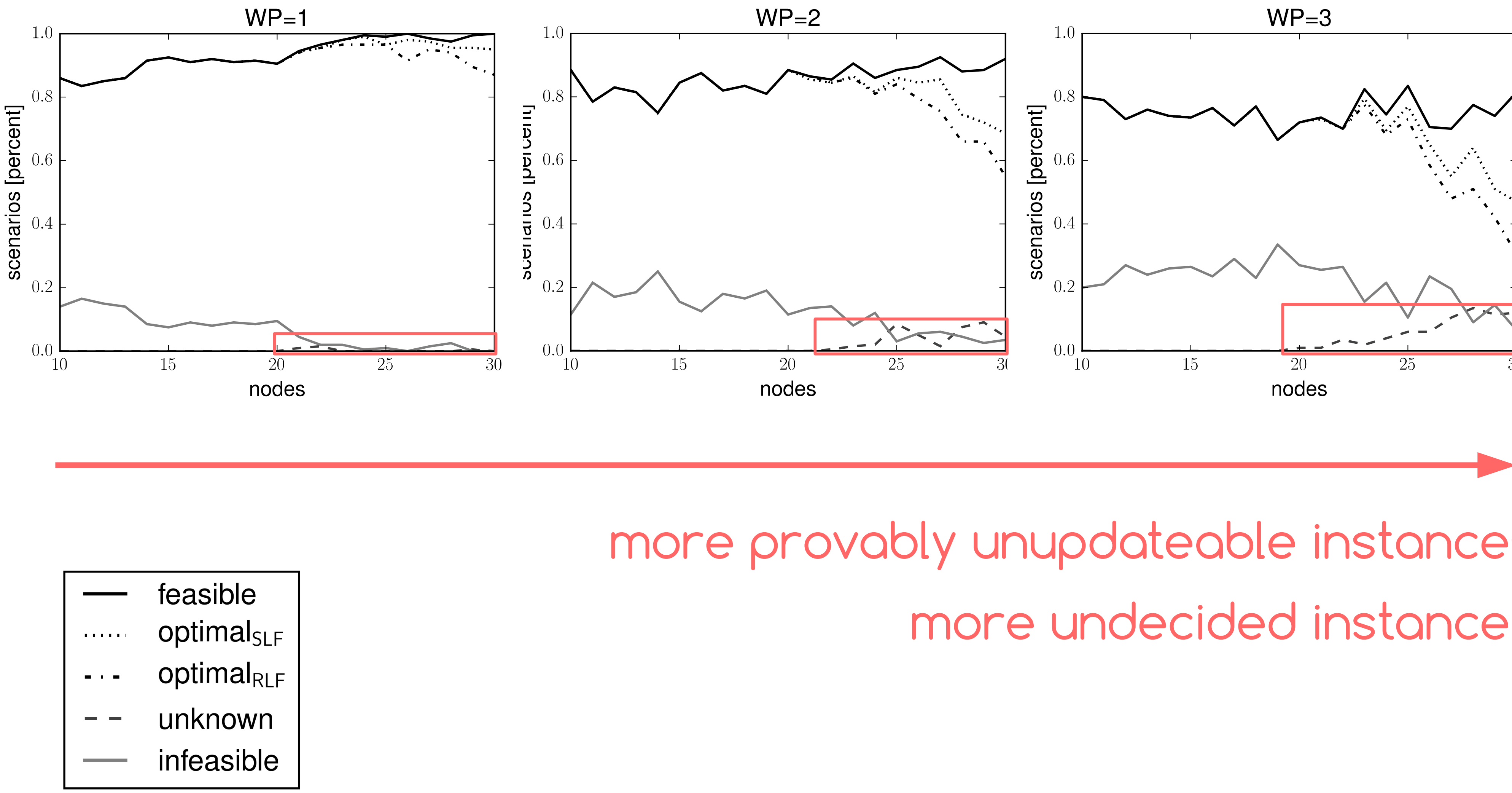
Computational Study: Solvability



more provably unupdateable instances



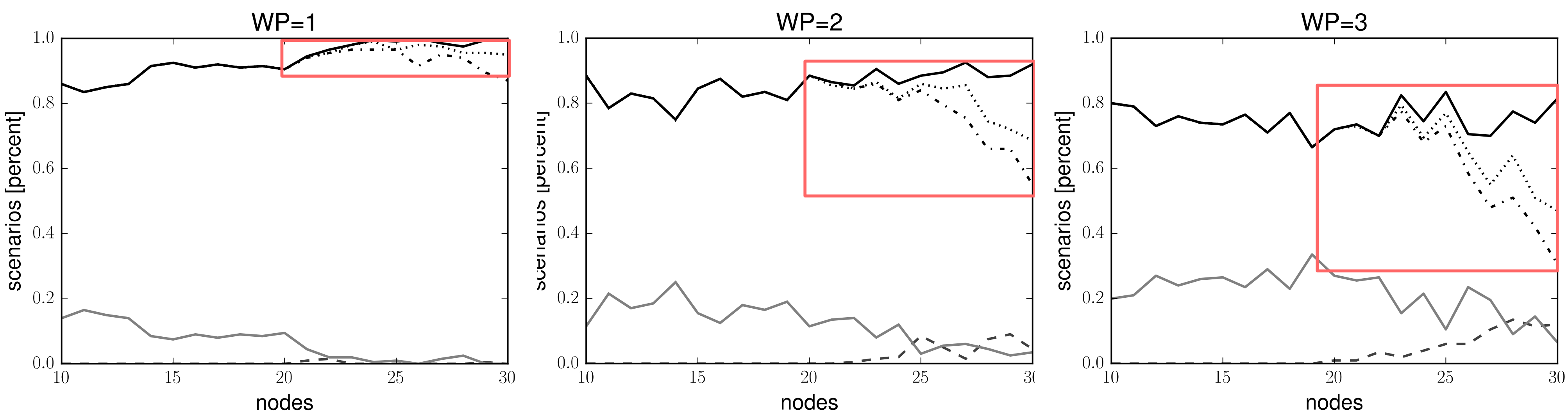
Computational Study: Solvability



more provably unupdateable instances

more undecided instances

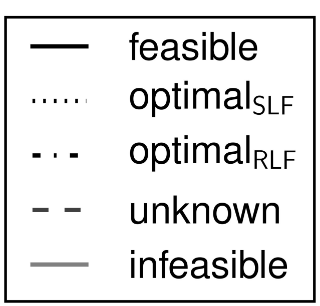
Computational Study: Solvability



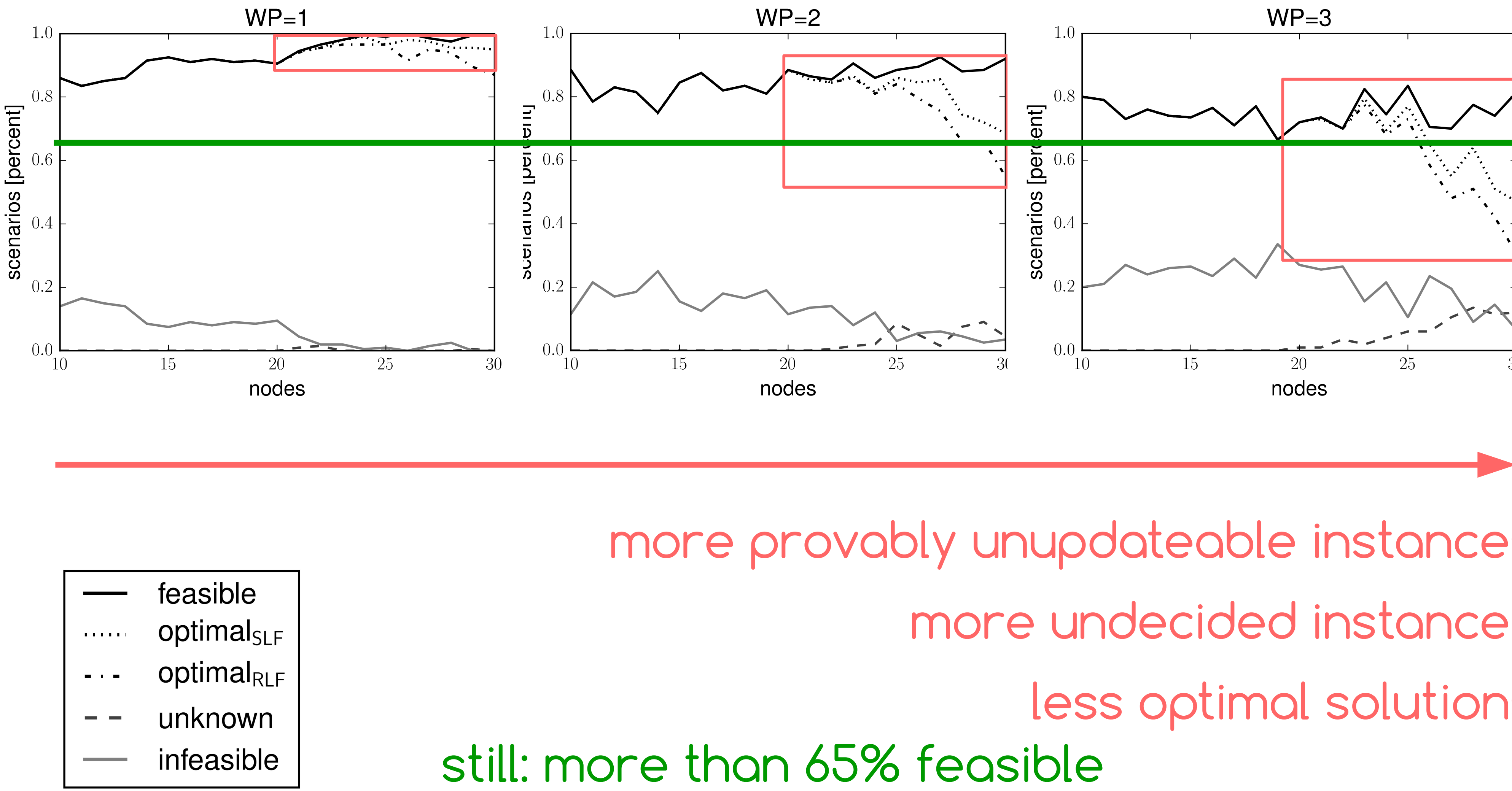
more provably unupdateable instances

more undecided instances

less optimal solutions



Computational Study: Solvability



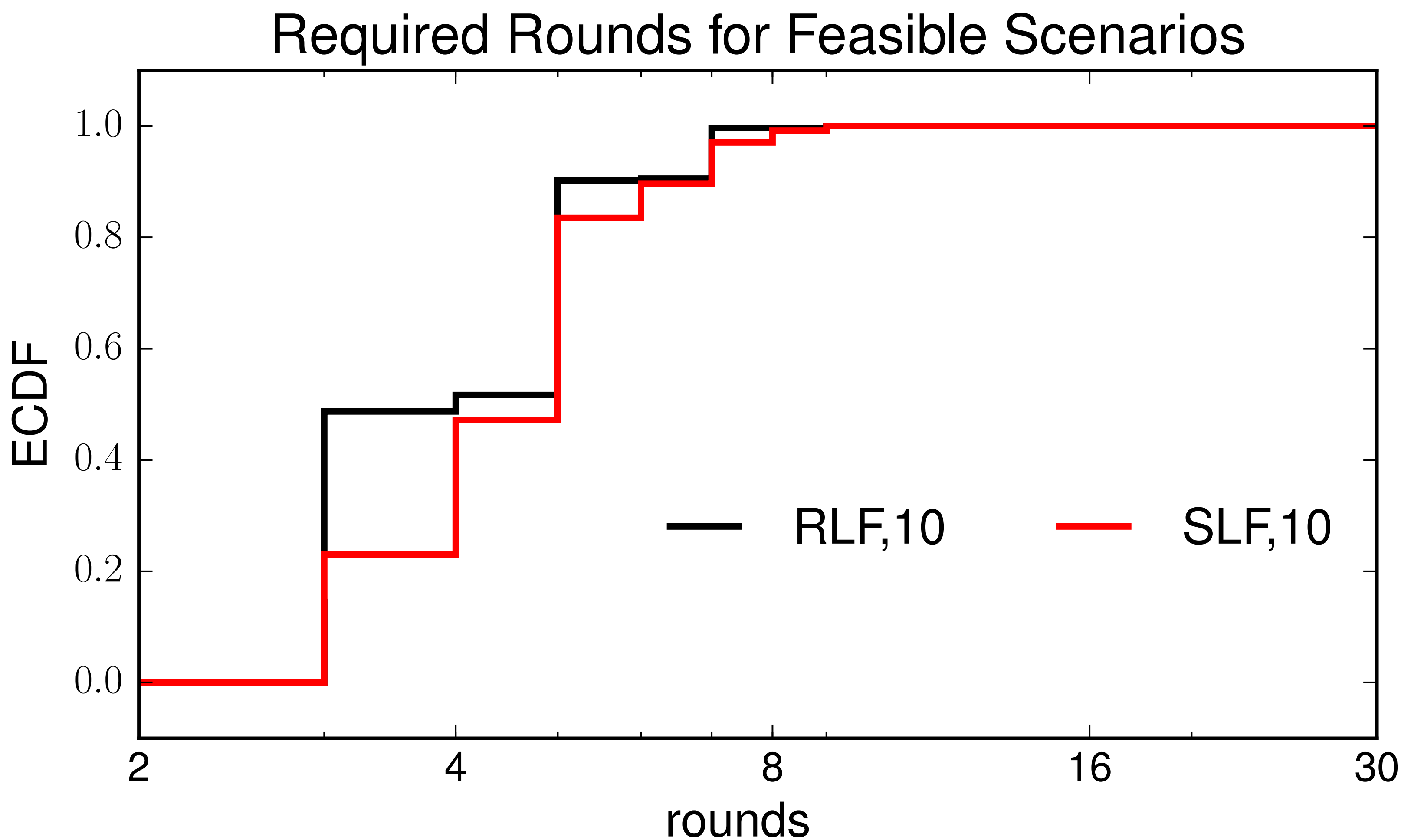
more provably unupdateable instances

more undecided instances

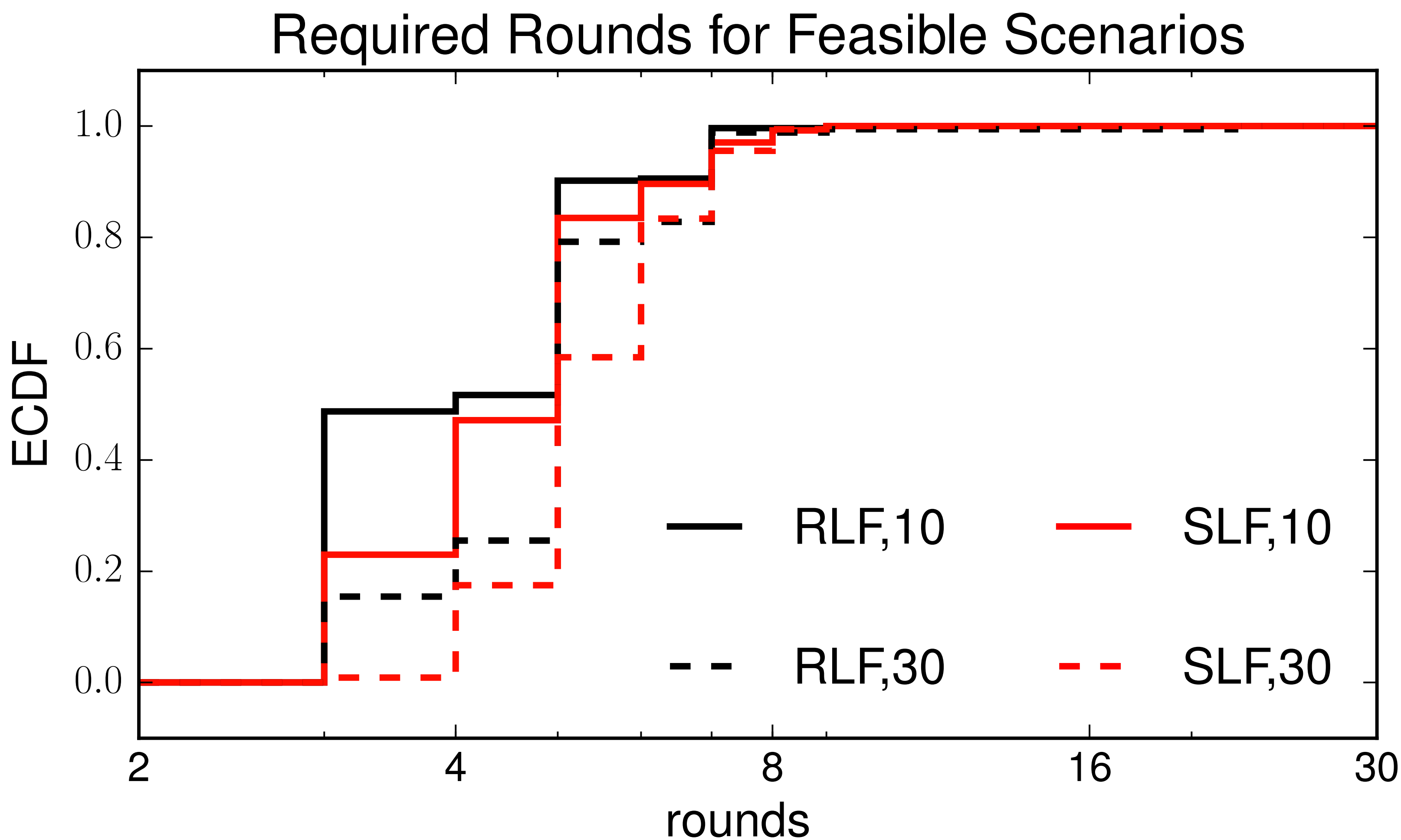
less optimal solutions

still: more than 65% feasible

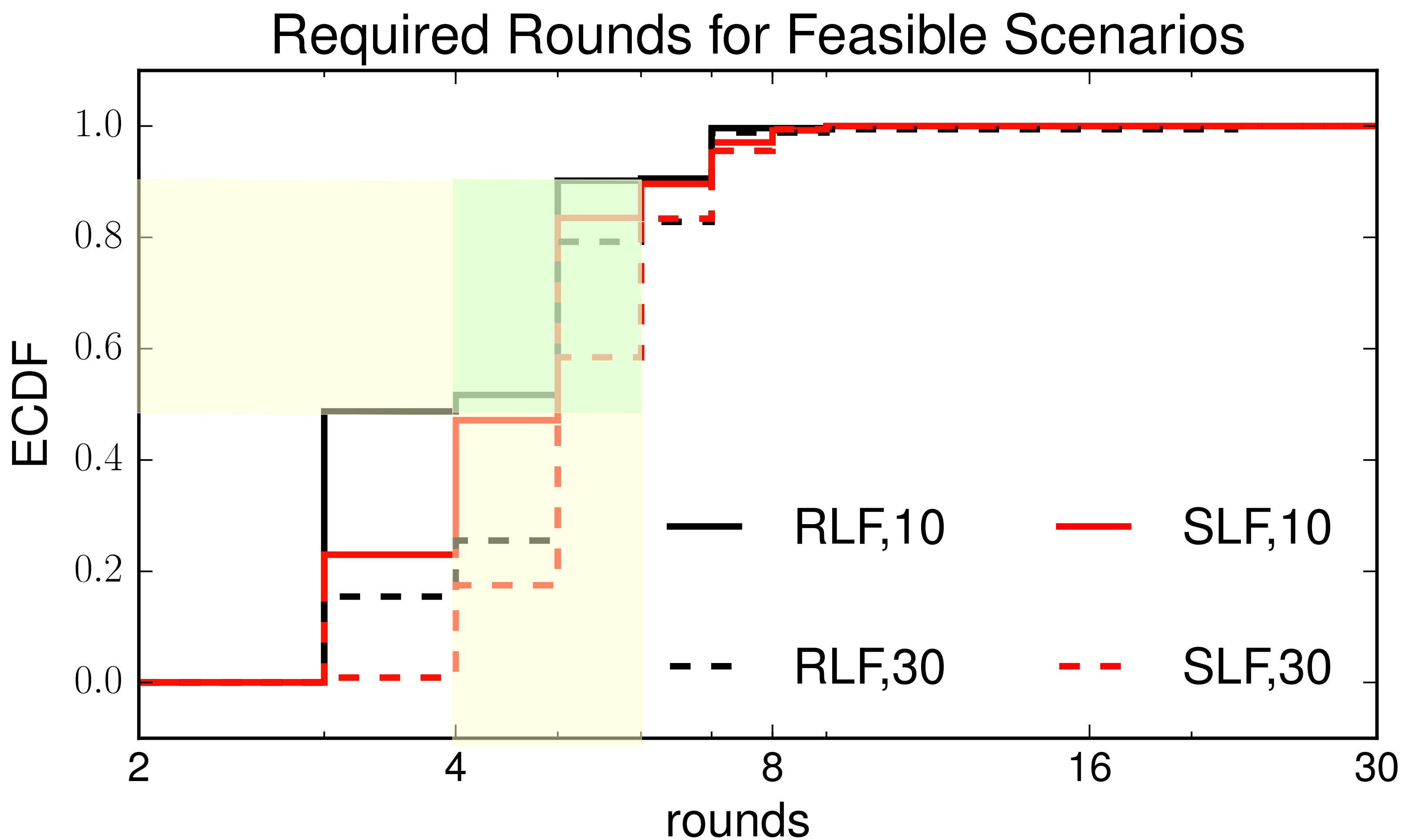
Computational Study: RLF vs. SLF



Computational Study: RLF vs. SLF



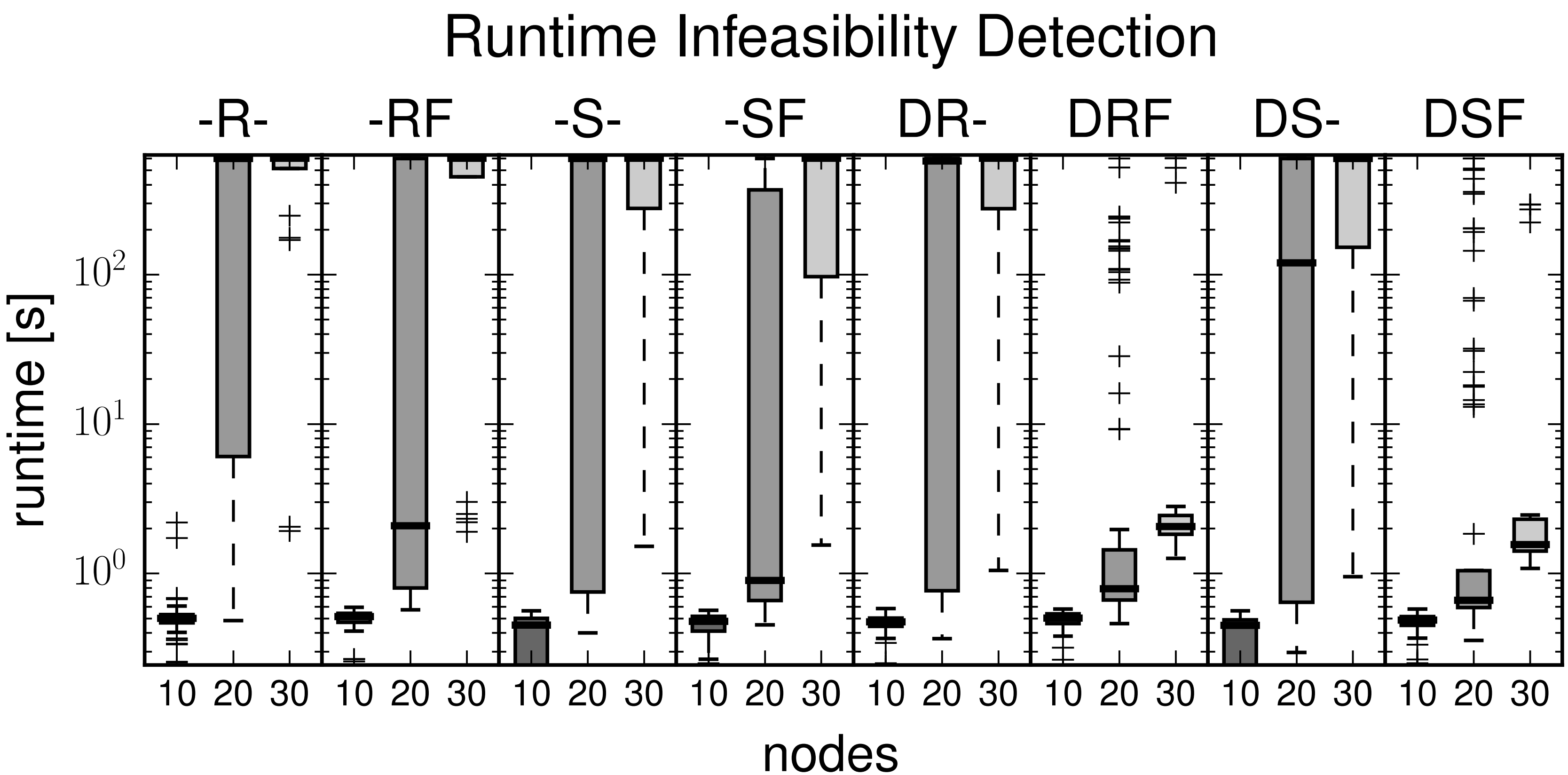
Computational Study: RLF vs. SLF



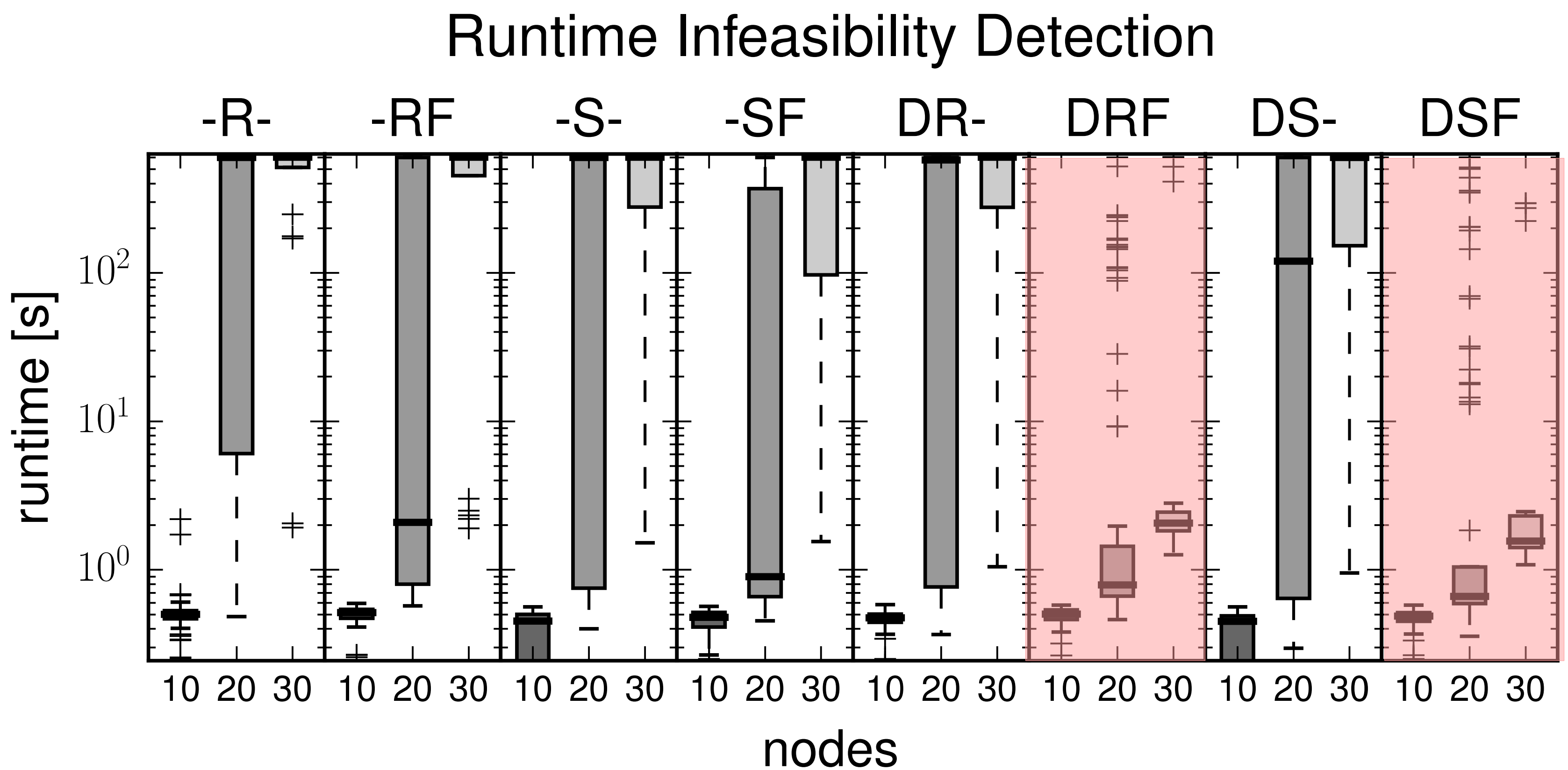
50% to 90% within 4-6 rounds

Computational Study: Formulation Performance

Computational Study: Formulation Performance

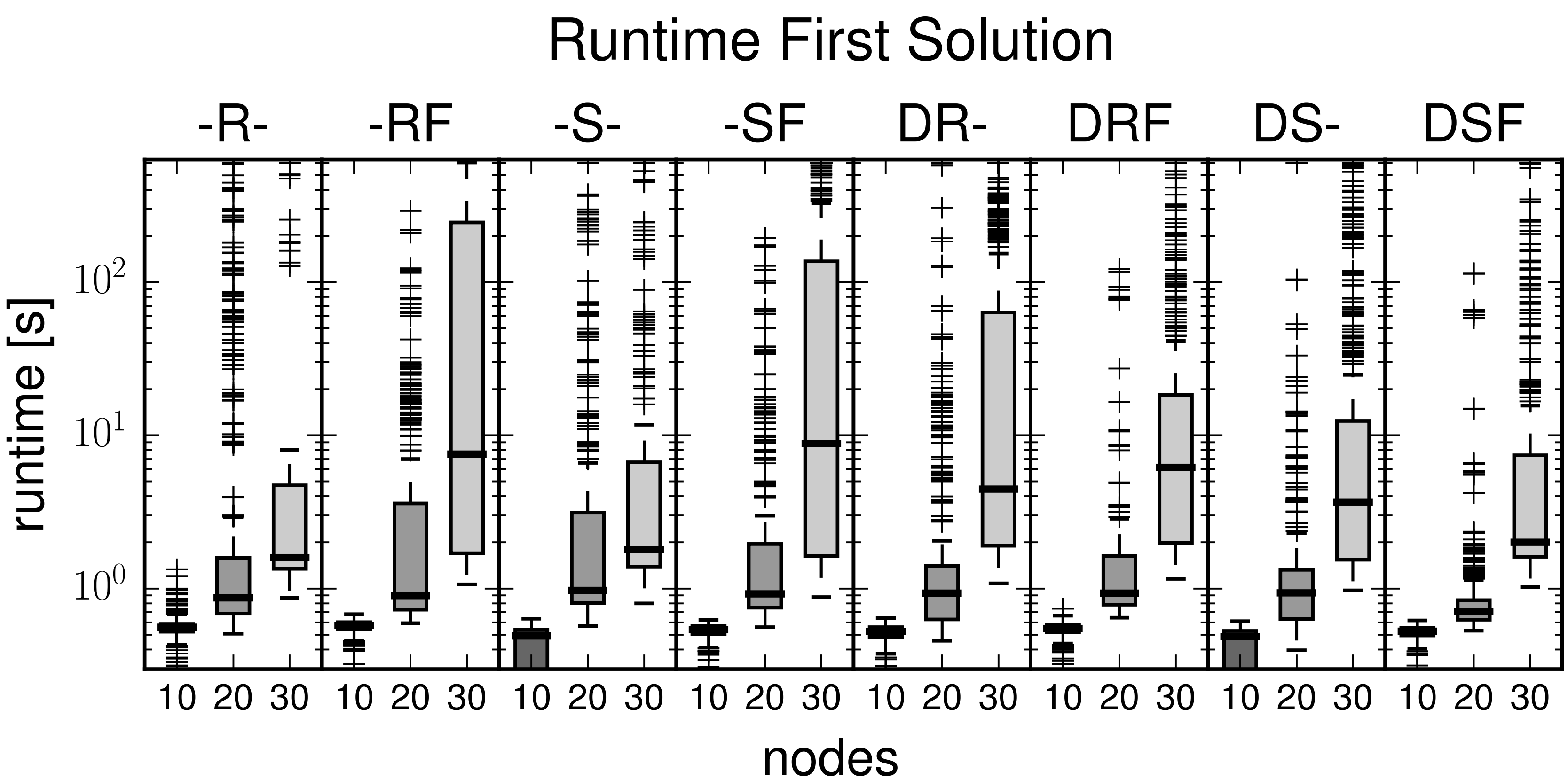


Computational Study: Formulation Performance

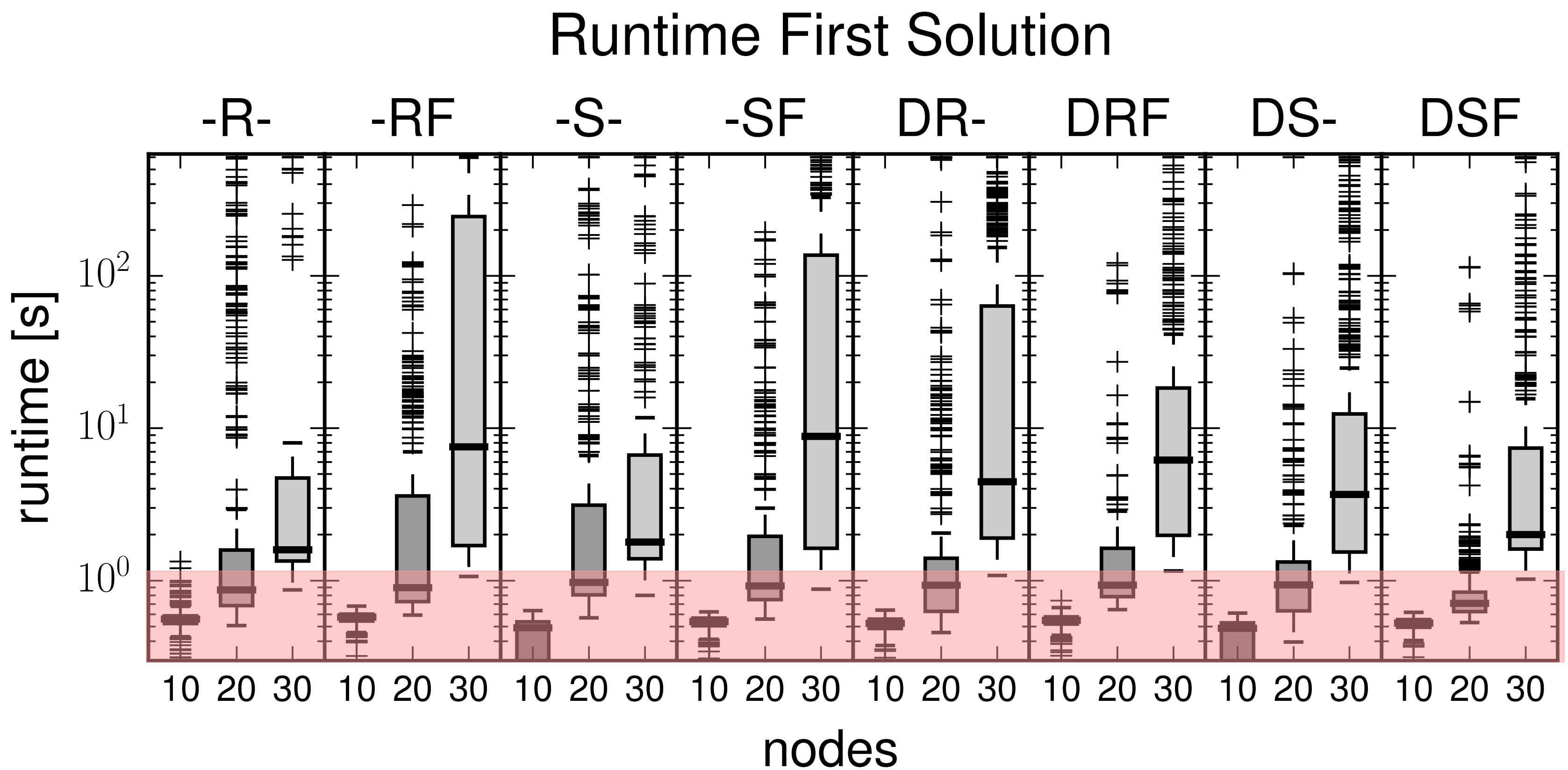


Combining Decision and Flow extension yields infeasibility certificates approx. 2 orders of magnitude faster.

Computational Study: Formulation Performance

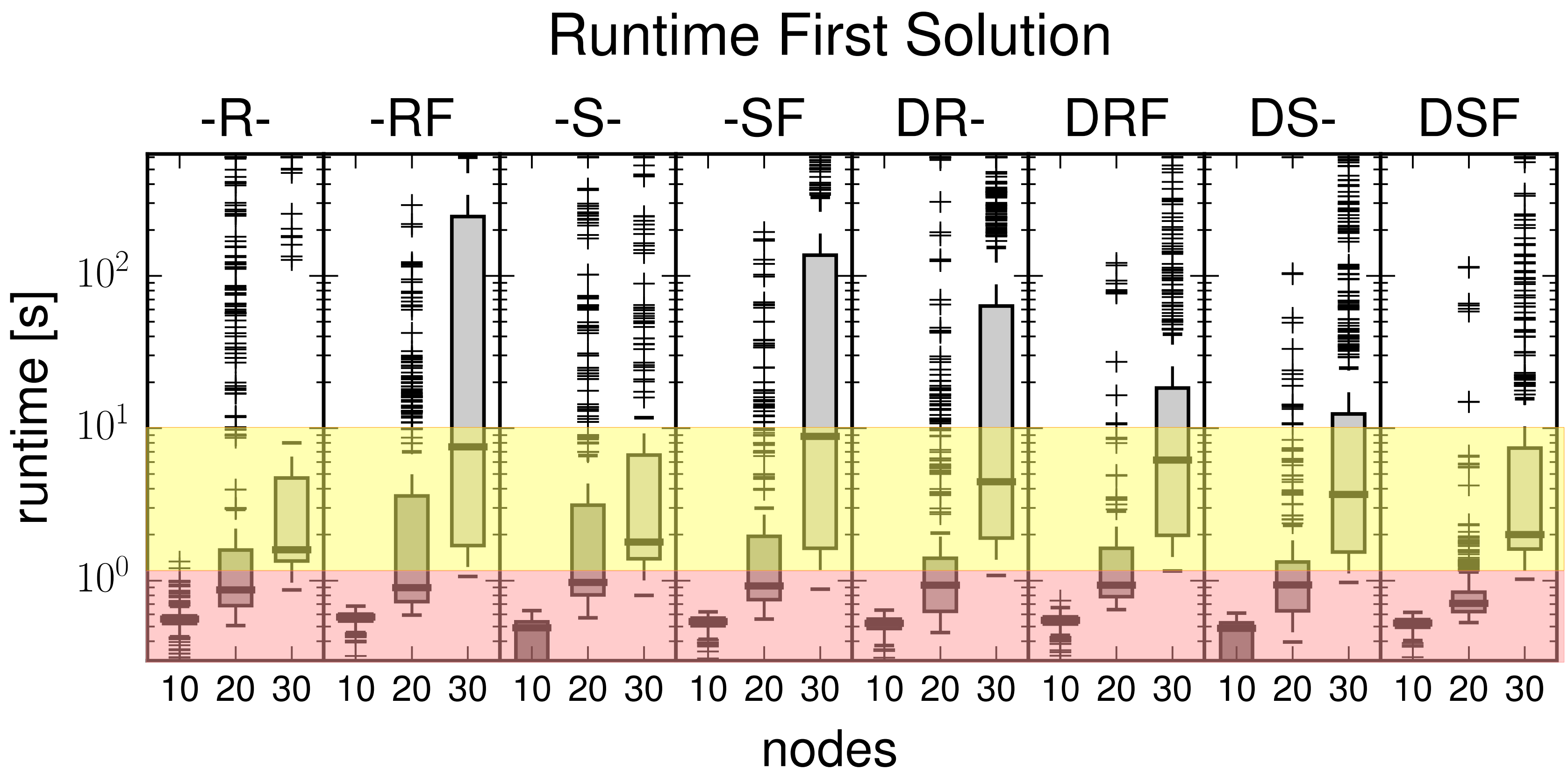


Computational Study: Formulation Performance



Median time for finding first solution:
< 1 second for 10 and 20 nodes

Computational Study: Formulation Performance



Median time for finding first solution:
< 1 second for 10 and 20 nodes
< 10 seconds for 30 nodes

Related Work

Loop Freedom

- Model and greedy algorithm [Mahajan et al., HotNets '13]
- NP-hardness of optimization, introduction of RLF [Ludwig et al., PODC '15]
- Updating multiple schedules at the same time [Dudycz et al., DSN '16 (to appear)]
- Hardness of computing maximum set of switches to update [Amiri et al., SIROCCO '16 (to appear)]

Waypoint Enforcement

- Introduction of WPE, impossibility and first MIP formulations [Ludwig et al., HotNets '14]

Conclusion

Problem

- Dynamic network updates ensuring LF and WPE

Theory

- LF + WPE may conflict
- LF + WPE is NP-hard to decide
- (other results)

Practice

- MIP Formulations for computing schedules
- Flow and Decision extensions to improve infeasibility detection

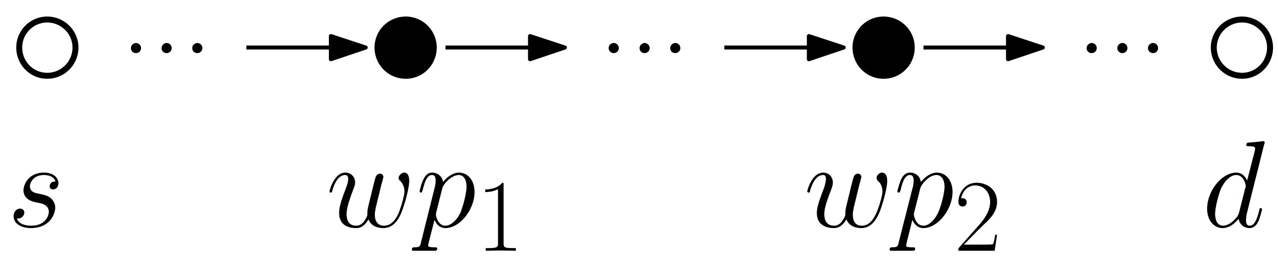
Evaluation

- Many scenarios are updateable using few rounds
- MIP formulations have reasonable runtimes

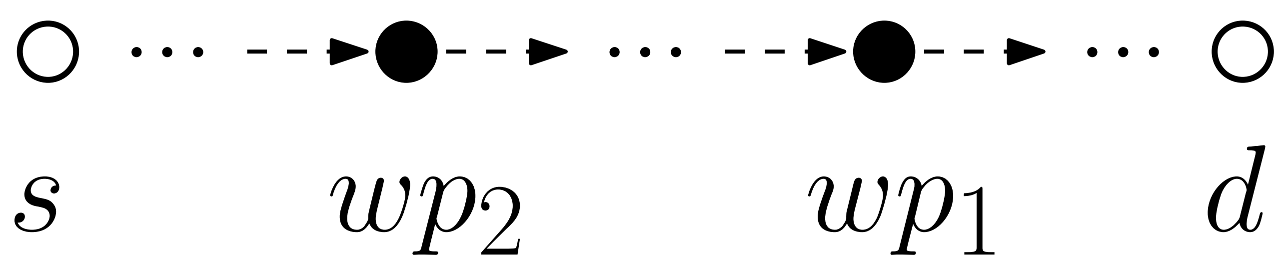
Backup

Theory:
Reordering Waypoints is impossible

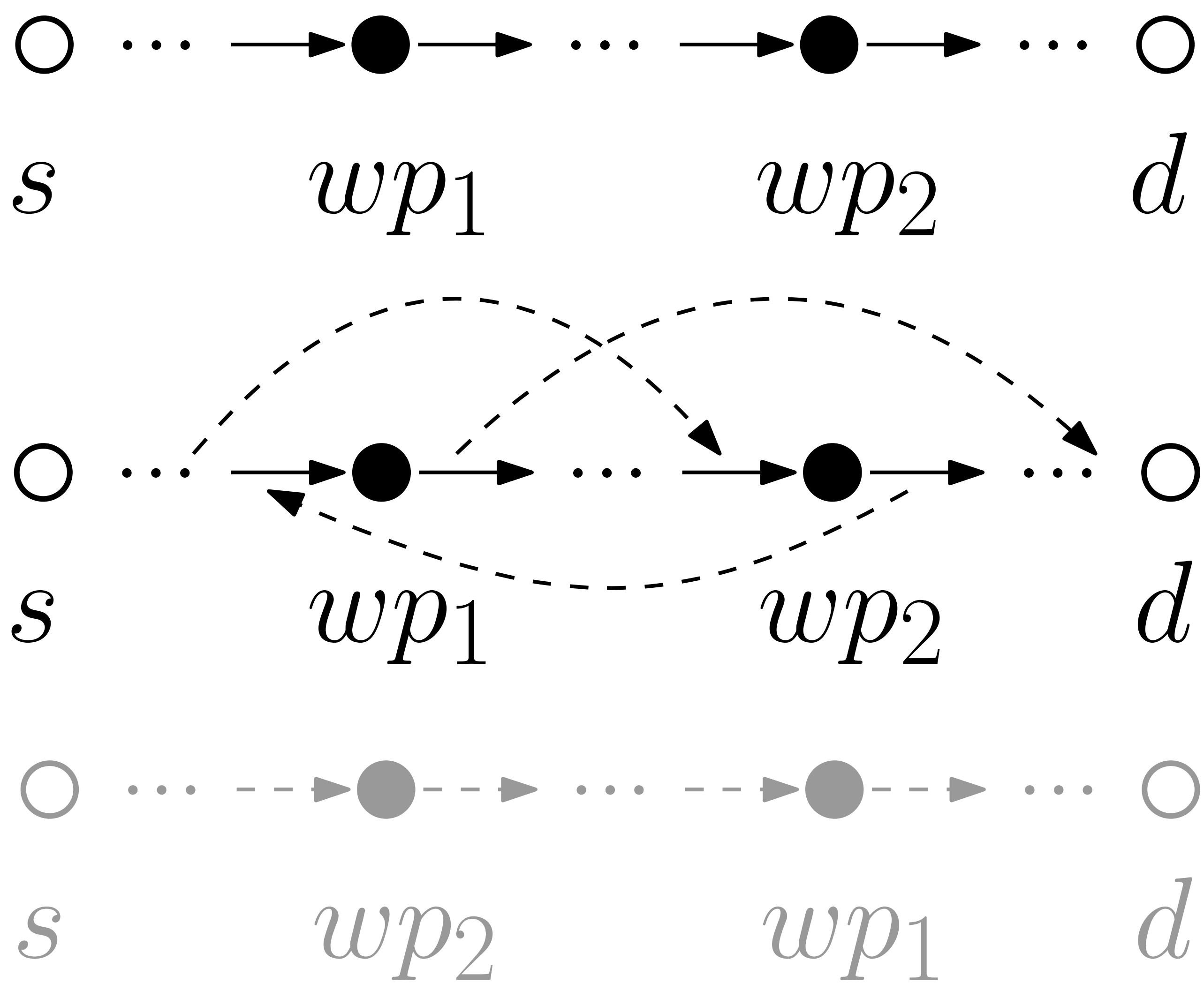
Reordering Waypoints is impossible



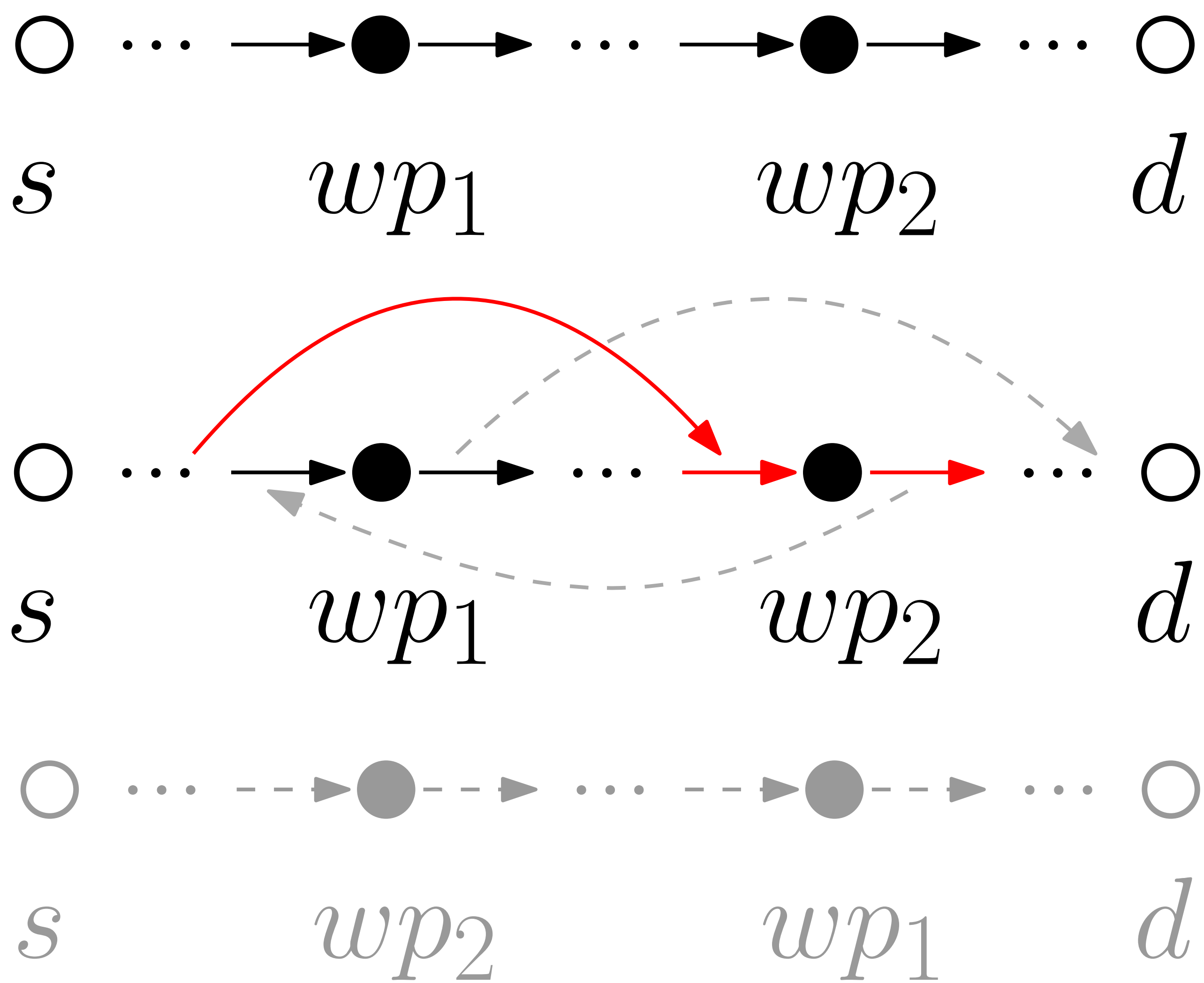
update to



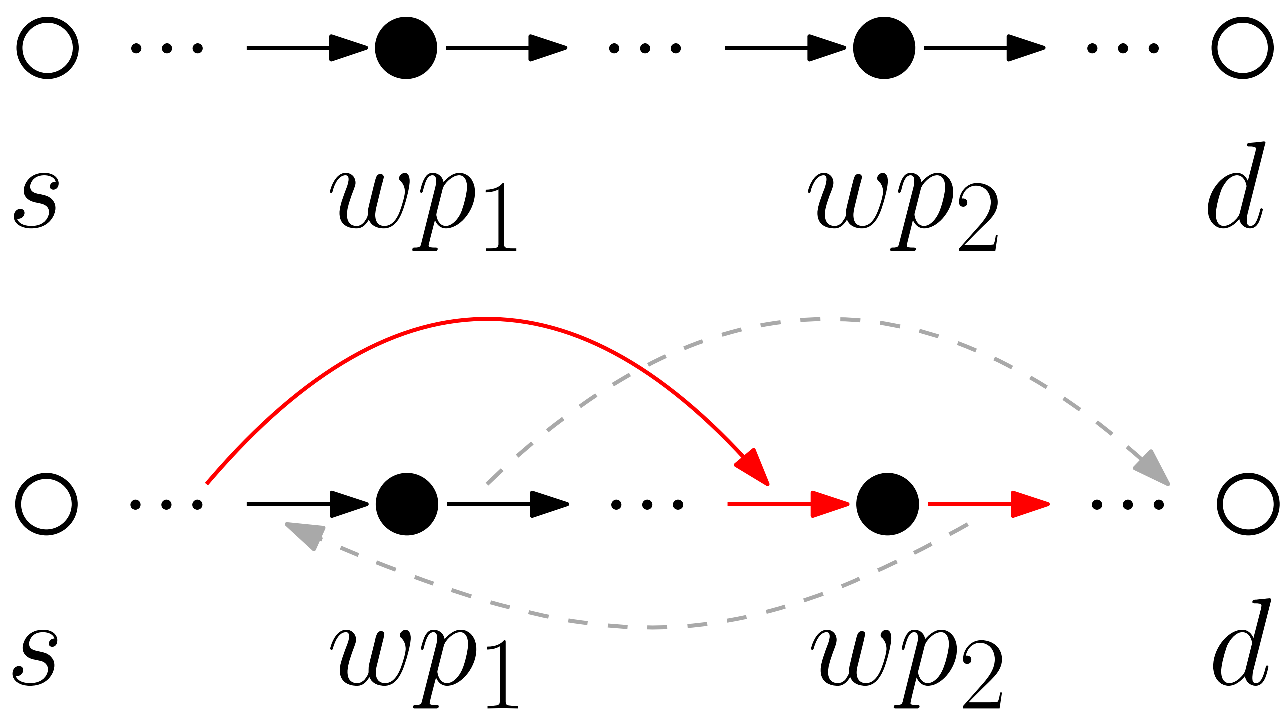
Reordering Waypoints is impossible



Reordering Waypoints is impossible



Reordering Waypoints is impossible

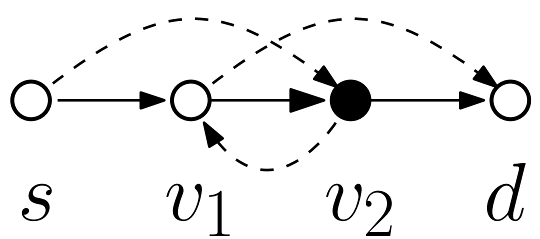


There must exist an update bypassing the first waypoint.

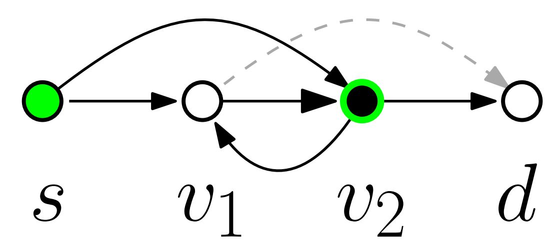
Theory:
WPE requires waiting

WPE requires waiting

State

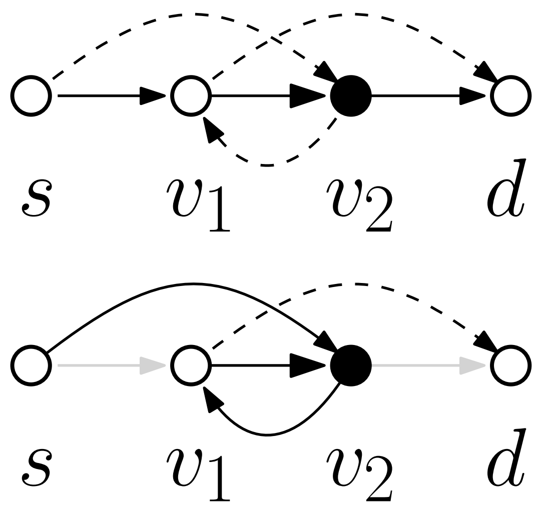


Temporary Forwarding Graph

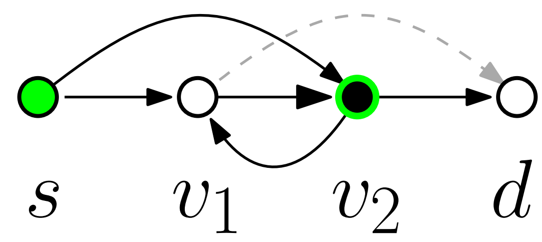


WPE requires waiting

State

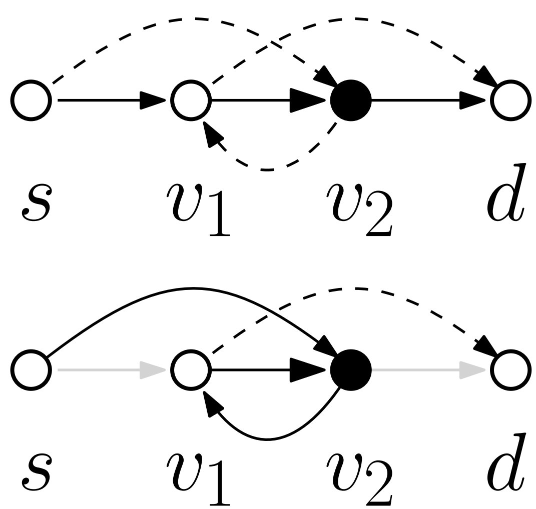


Temporary Forwarding Graph

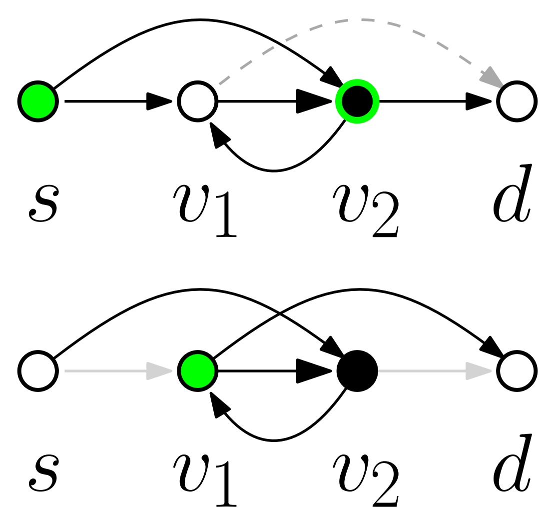


WPE requires waiting

State

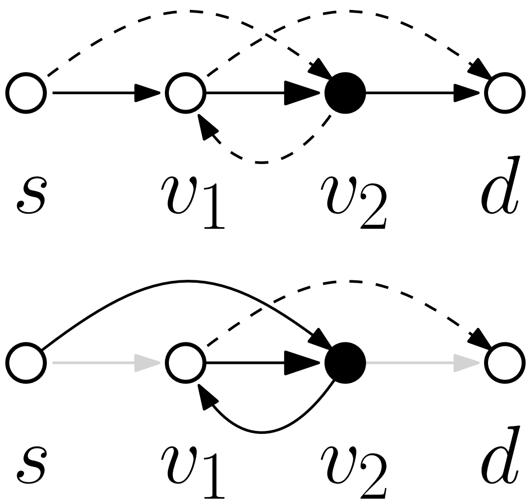


Temporary Forwarding Graph

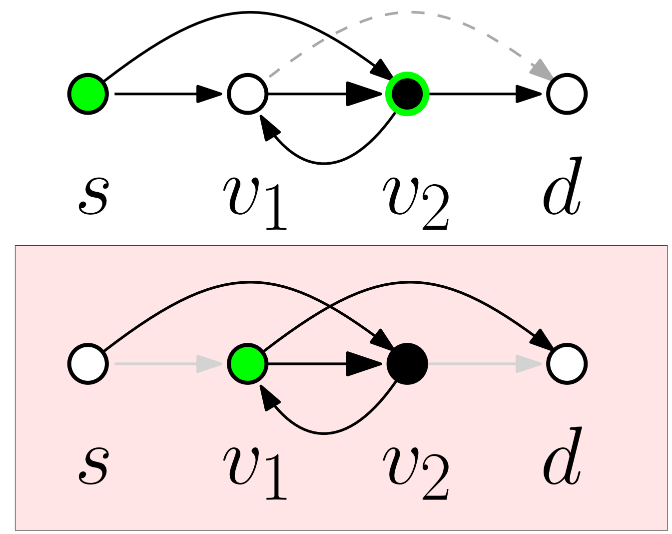


WPE requires waiting

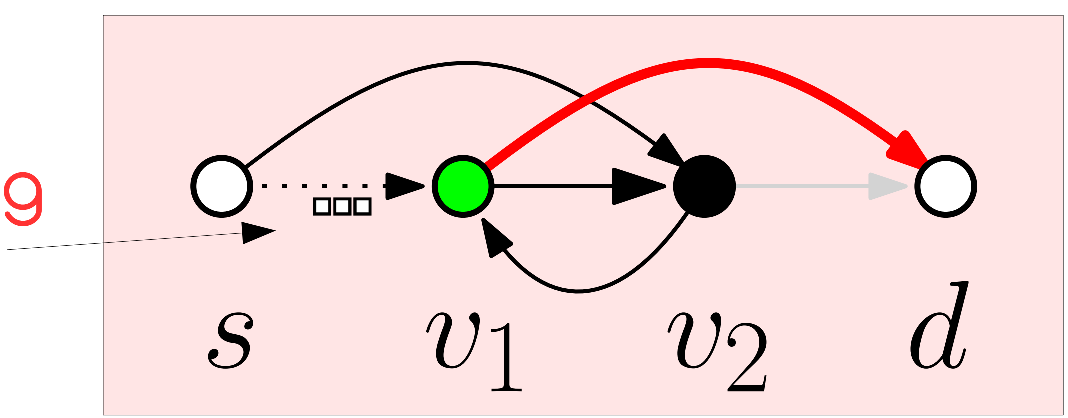
State



Temporary Forwarding Graph

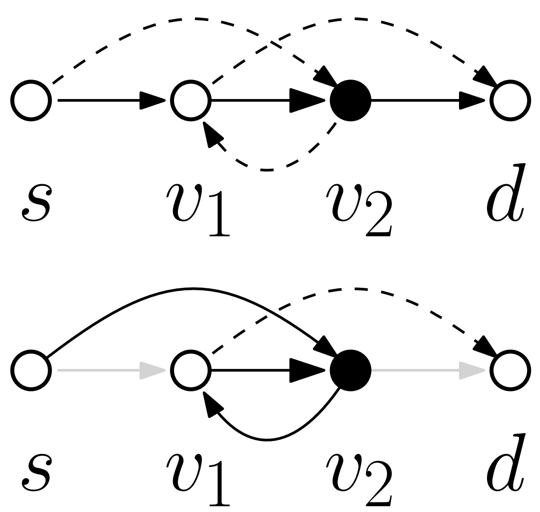


Packets still traversing link will bypass WP

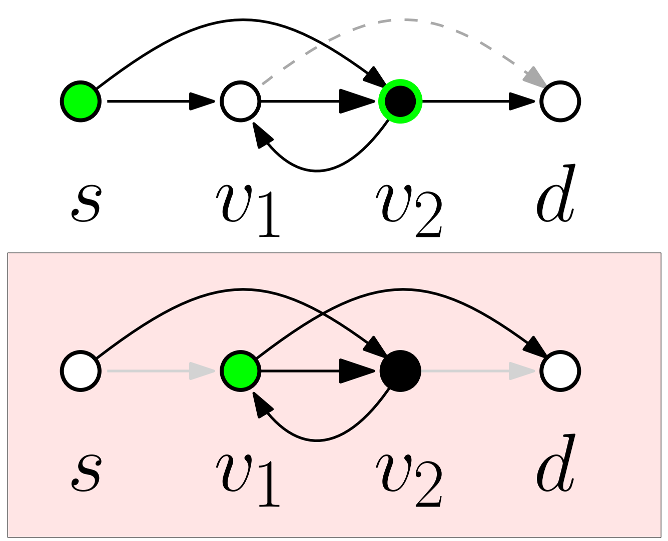


WPE requires waiting

State



Temporary Forwarding Graph



WPE requires upper bound on link delays, if the relative ordering of nodes changes.

Construction of 3-SAT Reduction: Remaining Connections