西安交通大學
XI'AN JIAOTONG UNIVERSITY

Fast SMT-Based Fault Tolerance Verification for **Wide Area Networks**

Ning Kang, Peng Zhang, Hao Li, Jianyuan Zhang | XJTU

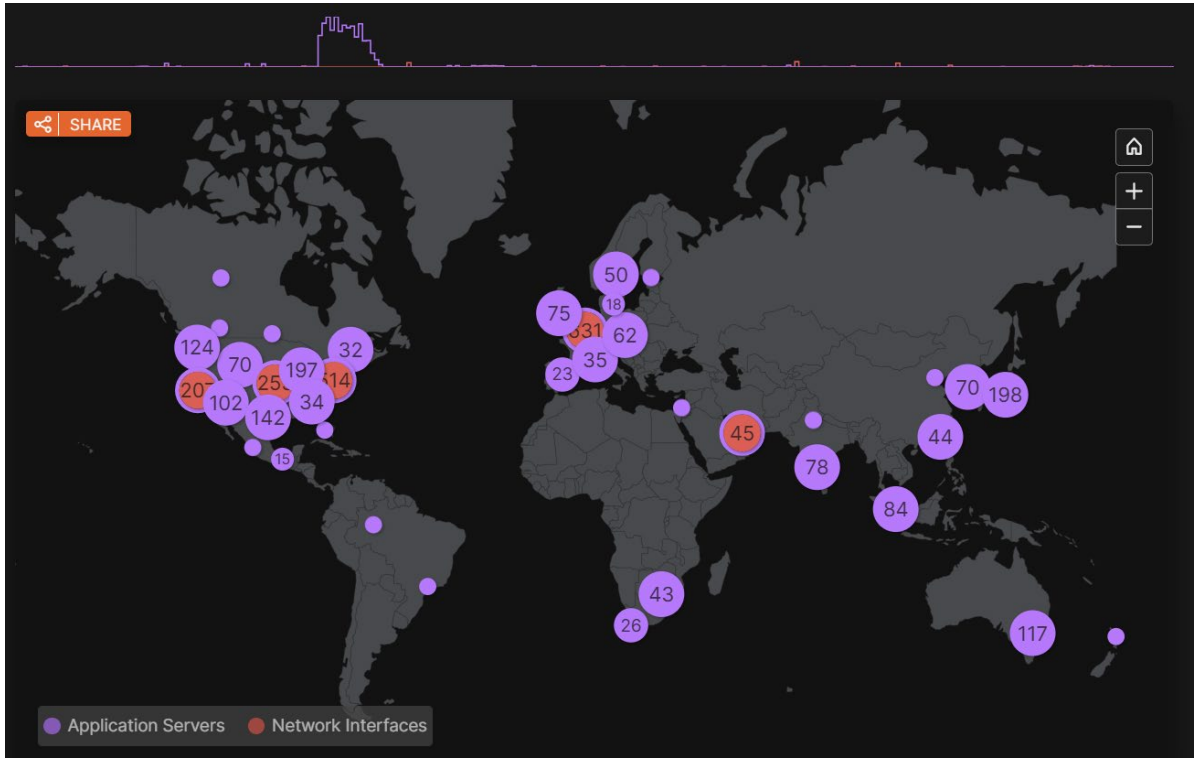
FM2026
The 27th International Symposium on Formal Methods
Tokyo, Japan
May 18 - 22, 2026

Network outages are common

- CISCO Thousandeyes

  One-Day Report
  (2026-04-24)

  Outages > 1k



https://www.thousandeyes.com/outages/

The main reasons are human errors

- CISCO Thousandeyes

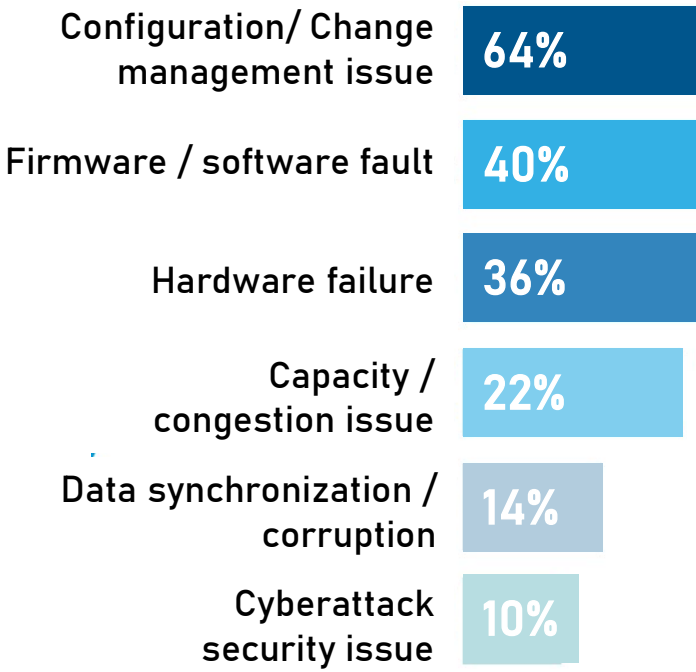
  One-Day Report
  (2026-04-24)

  Outages > 1k

- Uptime Institute

  Annual Report
  (2023)

  Human Error accounts for
  64% of outages



uptime INTELLIGENCE

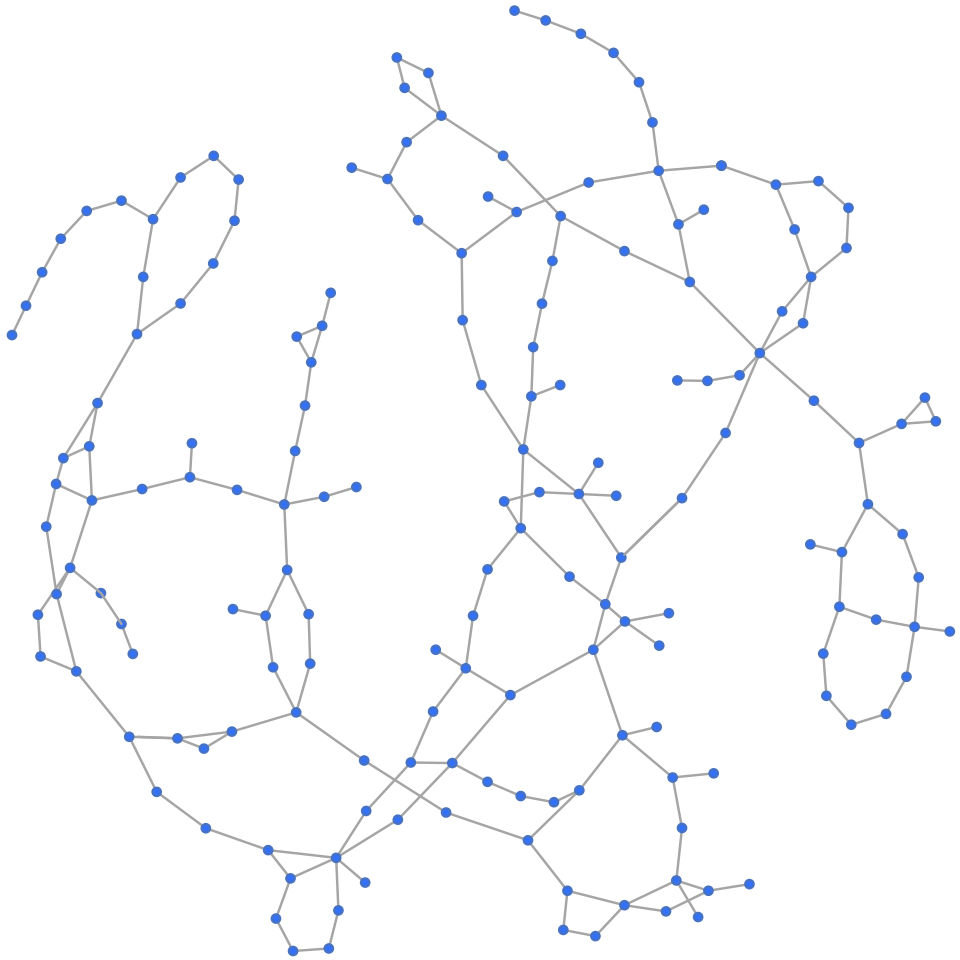
Uptime Instite's research points to one simple and actionable finding: Human error is often the result of failure to follow processes, or having inadequate processes. Better focus, management and training will produce better results.

https://www.datacenter-forum.com/uptime-institute/uptime-institutes-annual-outages-analysis-2023

Why are wide area networks error-prone?

- Irregular Topology

  No central nodes or hierarchical layers



Real-world Topology : US Carrier https://topology-zoo.org/

Why are wide area networks error-prone?

- Irregular Topology

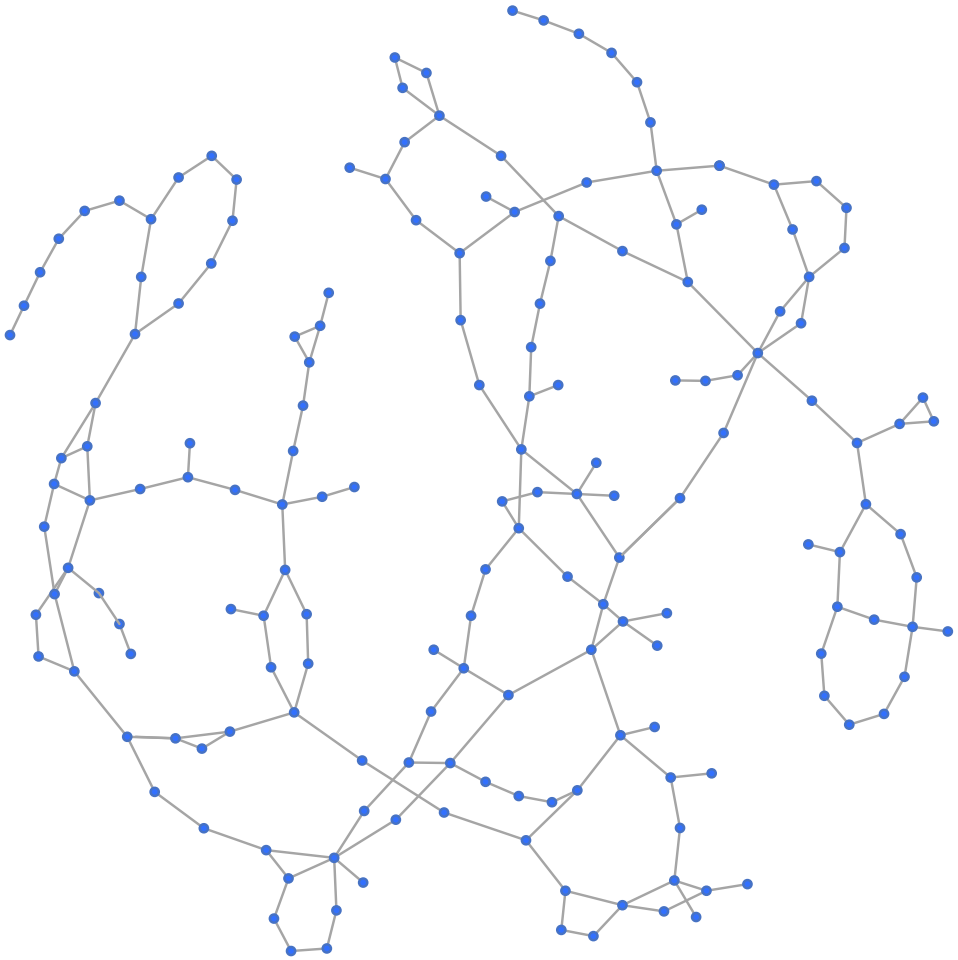
  No central nodes or hierarchical layers

- Complex Routing Policy

  > Router reflector

  > Route aggregation

  > BGP community ...



Real-world Topology : US Carrier https://topology-zoo.org/

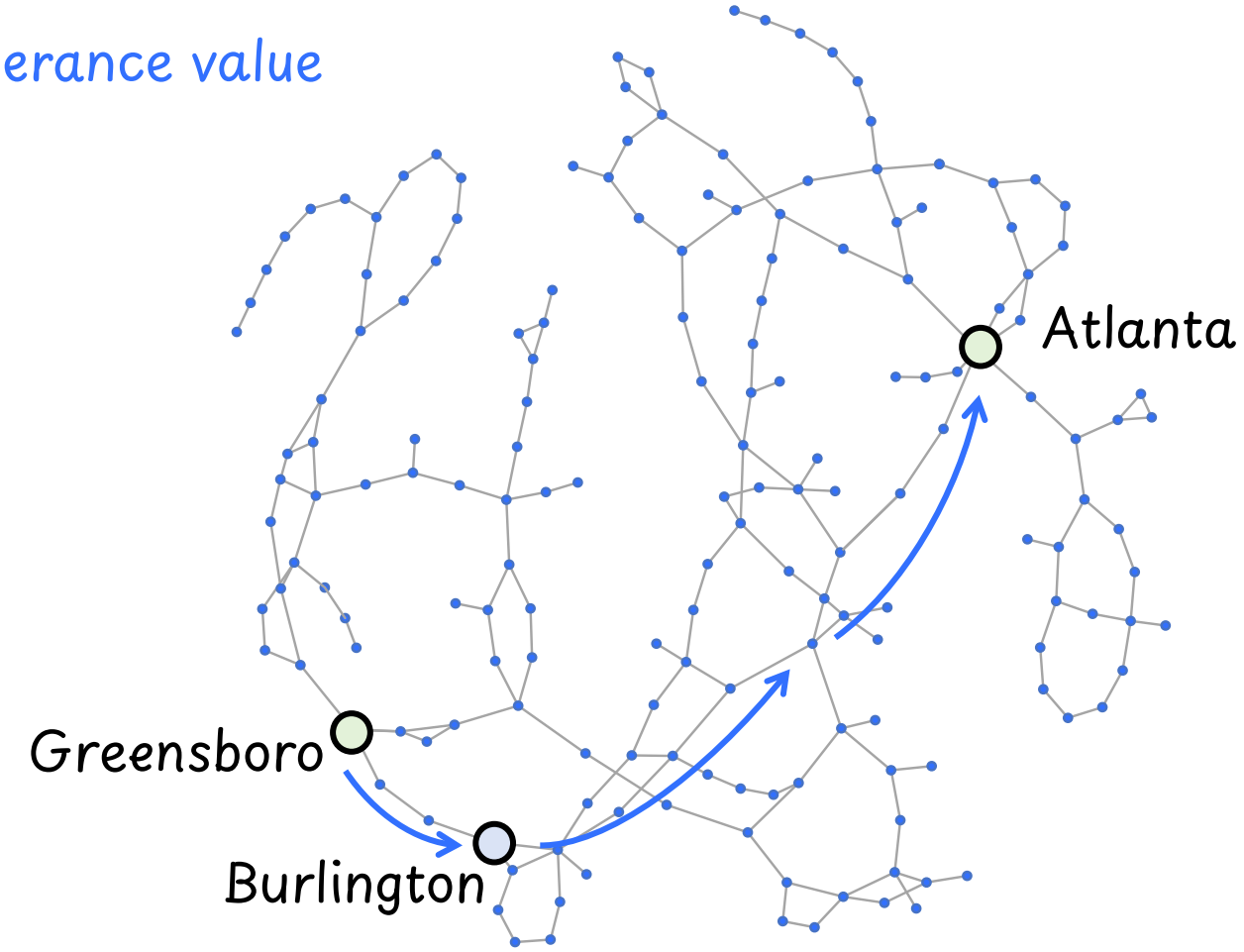
What does the operator want to know?

- **Reachability**

tolerance value

Greensboro->Atlanta: $k = 1$?

Can Atlanta still reach Greensboro even if any **one link fails**?



What does the operator want to know?

- **Reachability**

  tolerance value

  Greensboro->Atlanta: $k = 1$?

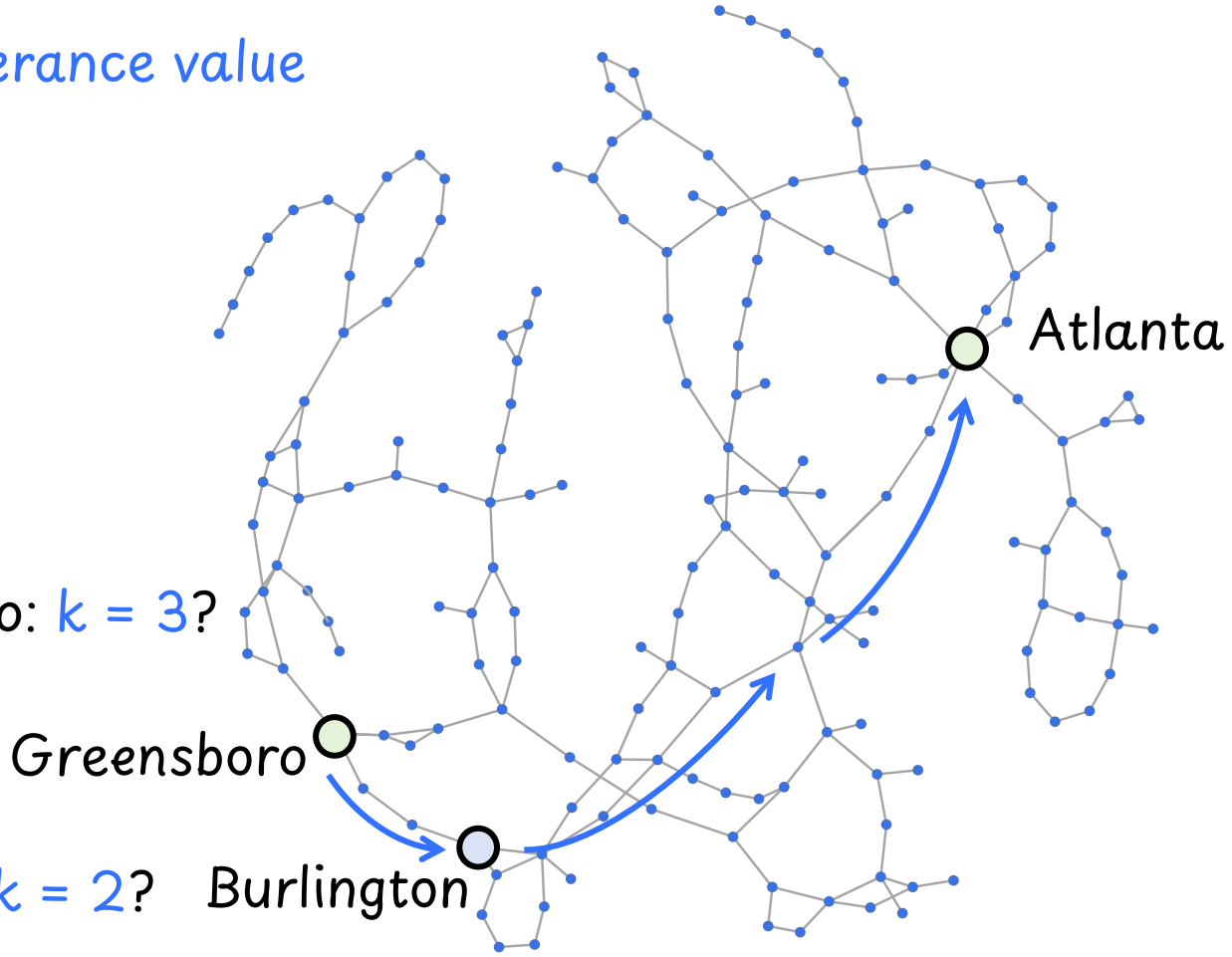
  Can Atlanta still reach Greensboro even if any **one link fails**?

- **Waypointing**

  Atlantam->Burlington->Greensboro: $k = 3$?

- **Load Balancing**

  Atlanta->Greensboro has 2 paths: $k = 2$?

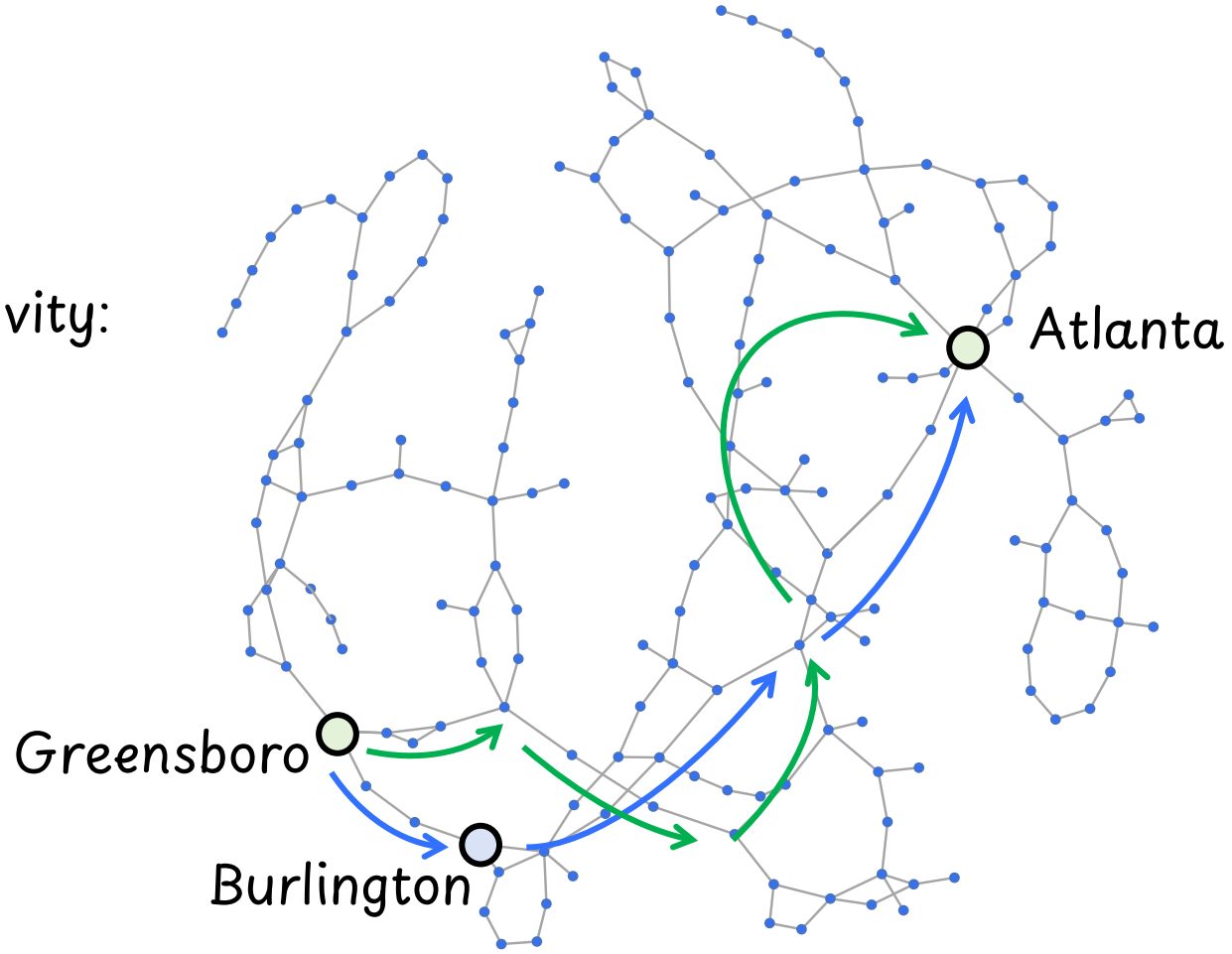


It is hard to know the tolerance value

- Reachability

  Greensboro->Atlanta: $k = 1$?

  - Only considering topology connectivity:
    The tolerance value $k = 1$



It is hard to know the tolerance value

- Reachability

Greensboro->Atlanta: $k = 1$?

- Only considering topology connectivity

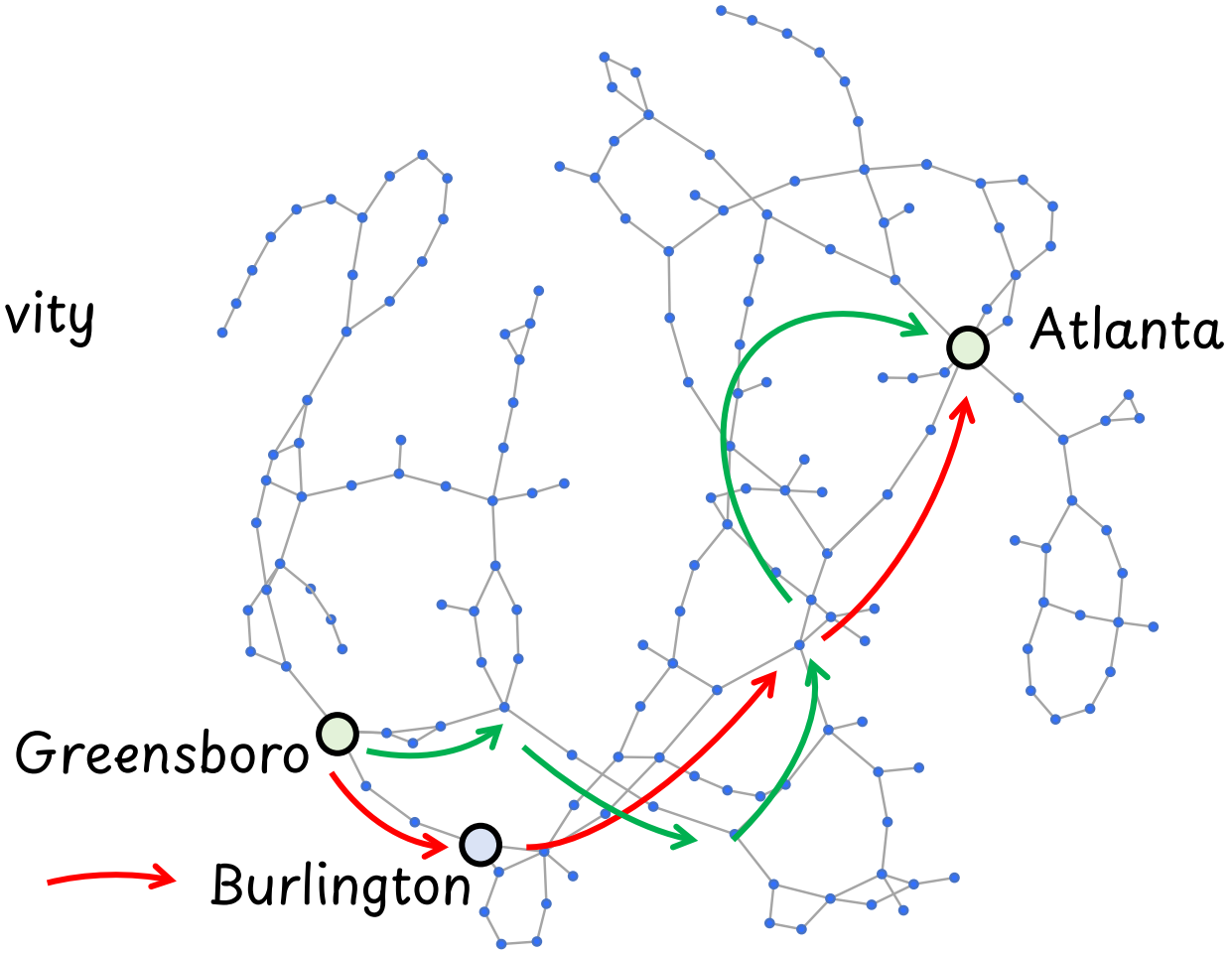
~~The tolerance value $k = 1$~~

- Considering routing policies

The tolerance value $k = 0$

```
ip prefix_list L permit 192.0.0.0/8 le 32
!
route-map v7_Import_From_v2 deny 10
  match ip address prefix-list L
!
route-map v7_Import_From_v2 permit 20
! match everything else
```

// Drops routes within 192.0.0.0/8



SMT-based verifiers can help operators

- Minesweeper [SIGCOMM'17]

SMT Constraints:

$$N \wedge \neg P \wedge K$$

Network Property Tolerance Value

- UNSAT ⇒ Property hold
- SAT ⇒ Property is violated

  SMT solution is counterexample

SMT-based verifiers can help operators

- Minesweeper [SIGCOMM'17]

SMT Constraints:

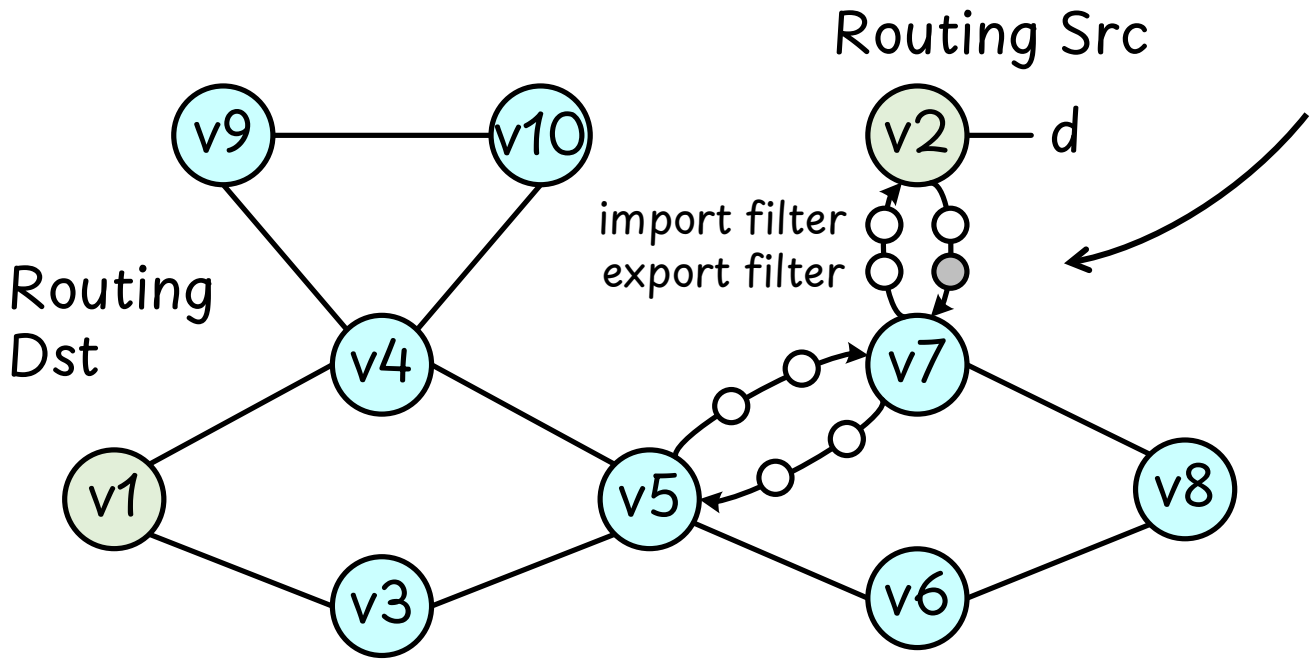
$$N \wedge \neg P \wedge K$$

Network

Encoding Network Constraints N

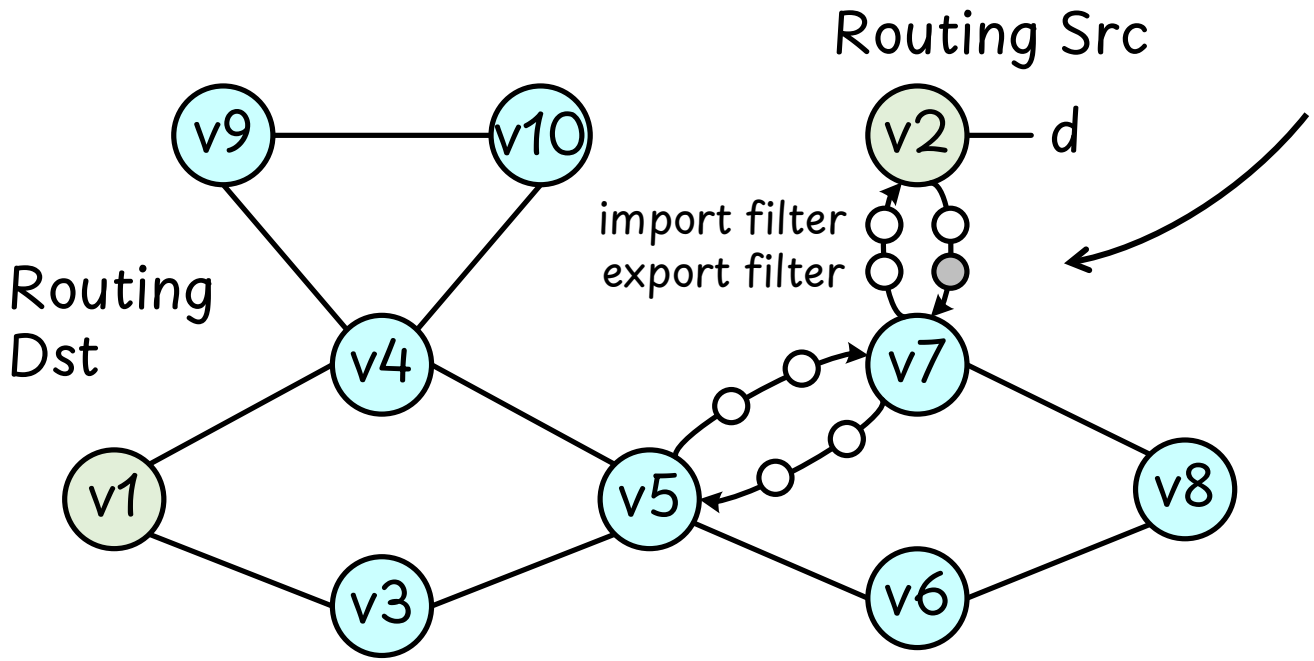
- Routing Propagation

```
ip prefix_list L permit 192.0.0.0/8 le 32
!
route-map v7_Import_From_v2 deny 10
  match ip address prefix-list L
!
route-map v7_Import_From_v2 permit 20
! match everything else
```



Encoding Network Constraints N

- Routing Propagation




```
ip prefix_list L permit 192.0.0.0/8 le 32
!
route-map v7_Import_From_v2 deny 10
  match ip address prefix-list L
!
route-map v7_Import_From_v2 permit 20
! match everything else
```

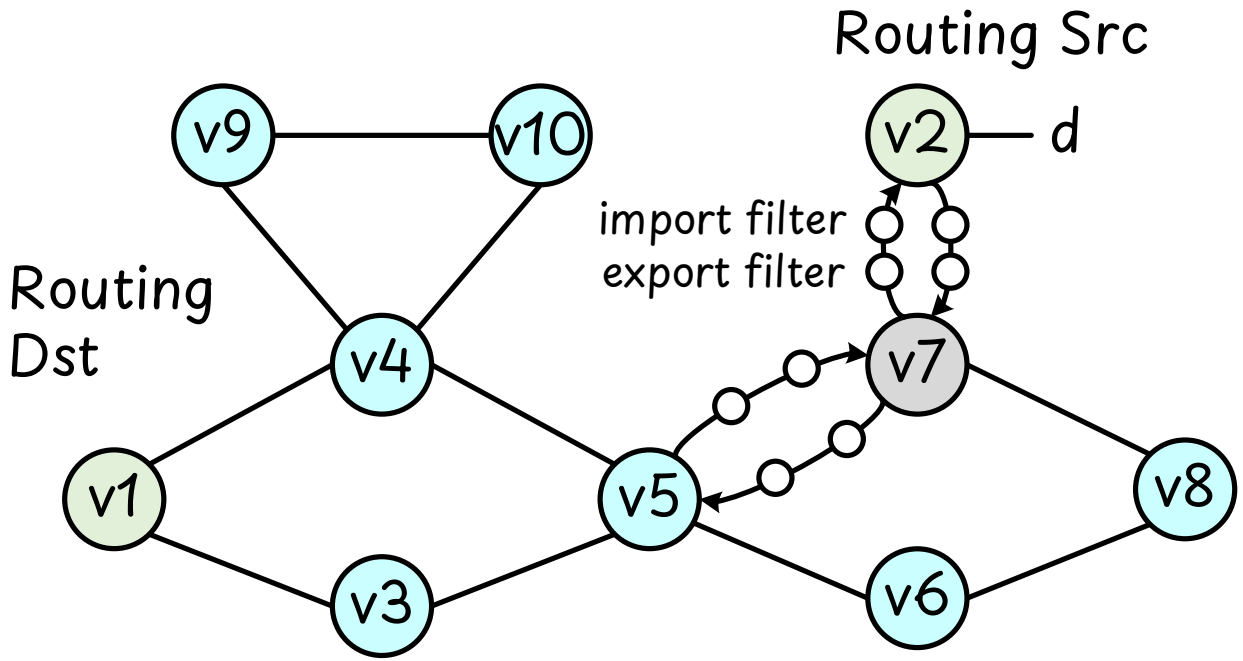
```
if e2.valid ∧ failedv2,v7 = 0 then
   if FBM(e2.prefix, 192.0.0.0, 8) ∧
     8≤ e2.prefixLen ≤ 32 then
      in7.valid = false
   else
      in7.valid = true
      in7.lp = e2.lp
      in7.prefixLen = e2.prefix
      ...
else
   in7.valid = false
```

SMT Variables

valid:	1 bit
prefix:	$[0,2^{32})$
prefixLen:	$[0,2^{5})$

Encoding Network Constraints N

- Routing Selection



The best route of V7: $best_{BGP}$

$$\bigwedge_{i \in \{2,5,8\}} best_{BGP} \preceq in_i \wedge \bigvee_{i \in \{2,5,8\}} best_{BGP} = in_i$$

SMT-based verifiers can help operators

- Minesweeper [SIGCOMM'17]

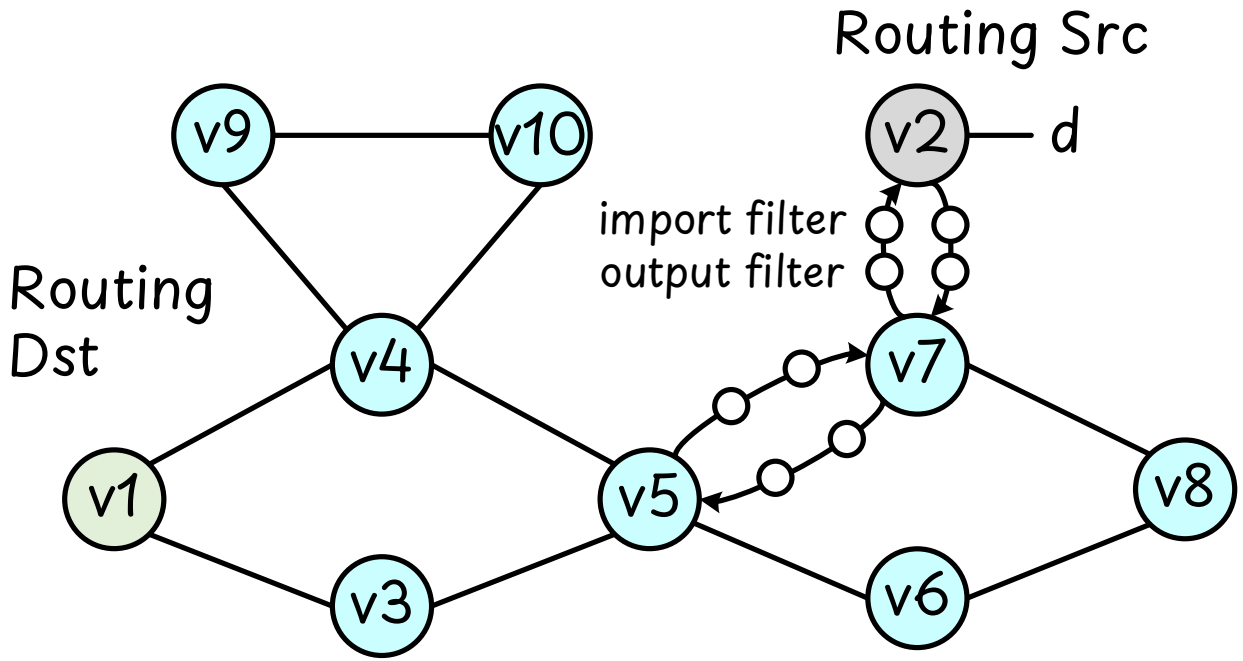
SMT Constraints:

$$N \wedge \neg P \wedge K$$

Property

Encoding Property Constraints P

Can router v1 (not) reach subnet d on router v2?

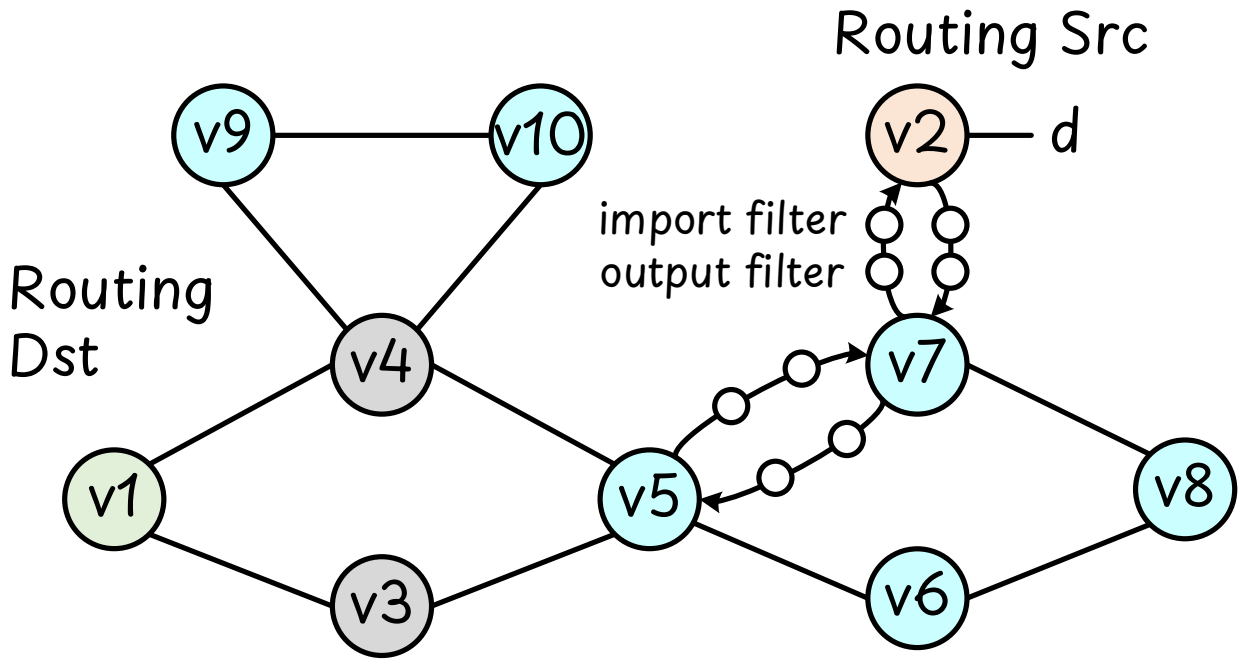


$canReach_{v2} \Leftrightarrow FIB_{v2,d}$

...

Encoding Property Constraints P

Can router v1 (not) reach subnet d on router v2?



$canReach_{v2} \Leftrightarrow FIB_{v2,d}$

...

$canReach_{v1} \Leftrightarrow$
$(FIB_{v1,v4} \wedge canReach_{v4}) \vee$
$(FIB_{v1,v3} \wedge canReach_{v3})$

Property: $canReach_{v1}$

Encoding Tolerance Value K

- Minesweeper [SIGCOMM'17]

SMT Constraints:

$$N \wedge \neg P \wedge K$$

Tolerance Value

$$K = \Sigma_{l \in L} f_l \leq k$$

The number of failed links is less than k

SMT-based verifiers exhibit low scalability

- Topology

  US Carrier Nodes: 158 Links: 189

- Large failure scenario space

  C(189, 3) ≈ O(10^6)

- Massive time overhead

  > Verifying a single property requires **hour-level** runtime (0.5 h) !!!

  > Verifying only 300 properties need about one week !!!

Tolerance Value Constraints

$$K = \Sigma_{l \in L} f_l \leq k$$

Can we verify a single property within a reasonable time (minute-level)?

Speedup Methods for SMT-Based Verifiers

- Reducing encoded variables — Bonsai TON'24, Origami CAV'19
- Adding constraints — BiNode TON'22, Trailblazer FM'23
- Guiding the solving process — NetSMT INFOCOM'23

Optimizing the solving process

Speedup Methods for SMT-Based Verifiers

- Reducing encoded variables — Bonsai TON'24, Origami CAV'19
- Adding constraints — BiNode TON'22, Trailblazer FM'23
- Guiding the solving process — NetSMT INFOCOM'23

Optimizing the solving process

- Modularizing the network — Timepiece PLDI'23, Kirigami TON'24, Lightyear SIGCOMM'23
- Ignoring protocol details — ACORN FMCAD'22

Changing the network model

Speedup Methods for SMT-Based Verifiers

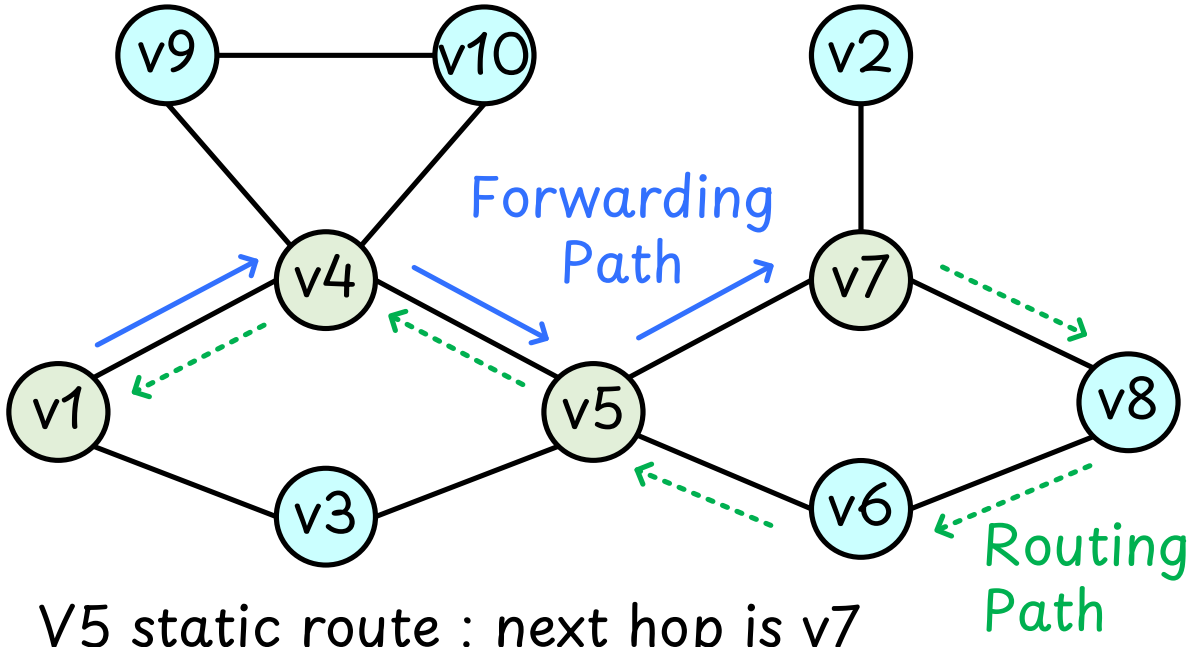
- Reducing encoded variables — Bonsai TON'24, Origami CAV'19 — Symmetric Network (e.g. DCN)
- Adding constraints — BiNode TON'22, Trailblazer FM'23 — Strong assumptions (e.g. Gao-Rexford principle)
- Guiding the solving process — NetSMT INFOCOM'23 — Specific properties (e.g. property is violated)

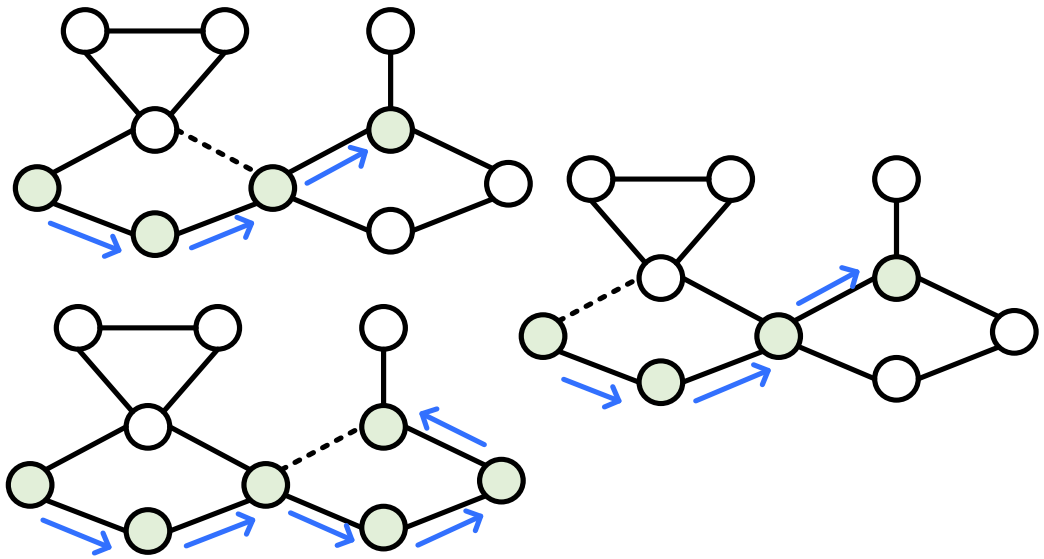
- Modularizing the network — Timepiece PLDI'23, Kirigami TON'24, Lightyear SIGCOMM'23
- Ignoring protocol details — ACORN FMCAD'22 ← Lose the ability to verify fault tolerance property

Limitation of Trailblazer

- Failing the links in forwarding paths

Consider reachability from v1 to v7 -> The tolerance value is k=1



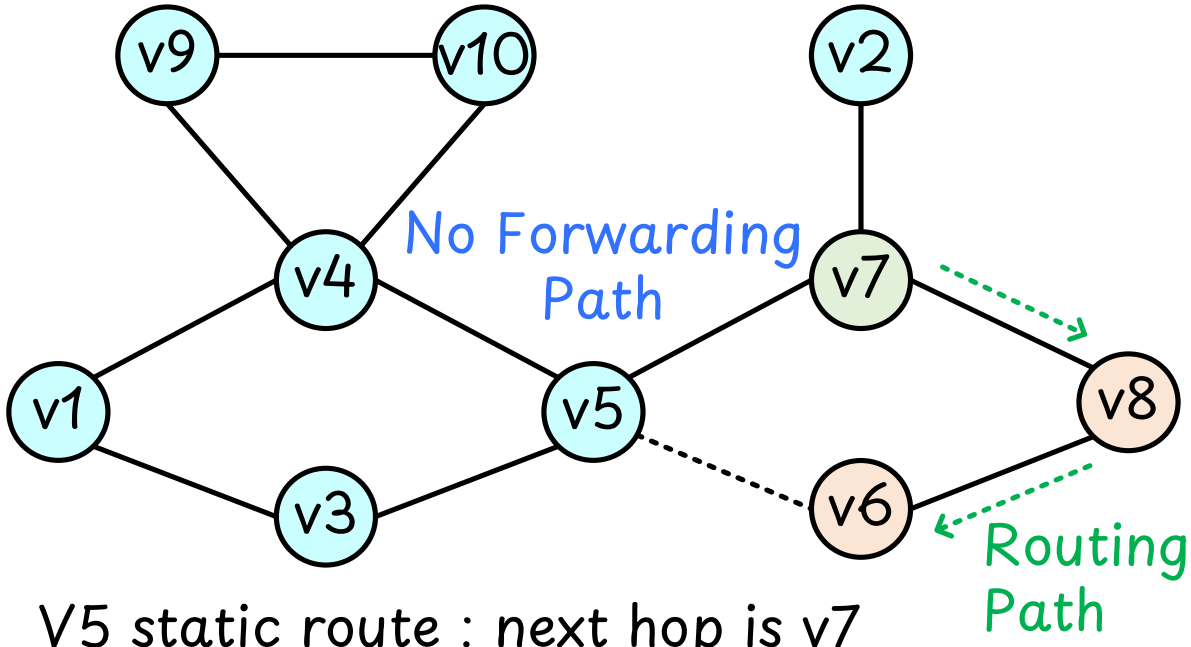


V5 static route : next hop is v7
V5 routing policy : drops routes from v7

Limitation of Trailblazer

- Failing the links in forwarding paths

Consider reachability from v1 to v7



V5 static route : next hop is v7
V5 routing policy : drops routes from v7

~~-> The tolerance value is k=1~~

-> The tolerance value is k=0

[Failing link v5-v6, the property is violated]

Trailblazer may produce false positives

We propose VeriBoost

without assuming regular topology structure or

specific routing policy

Basic Ideas - Reducing Links Space

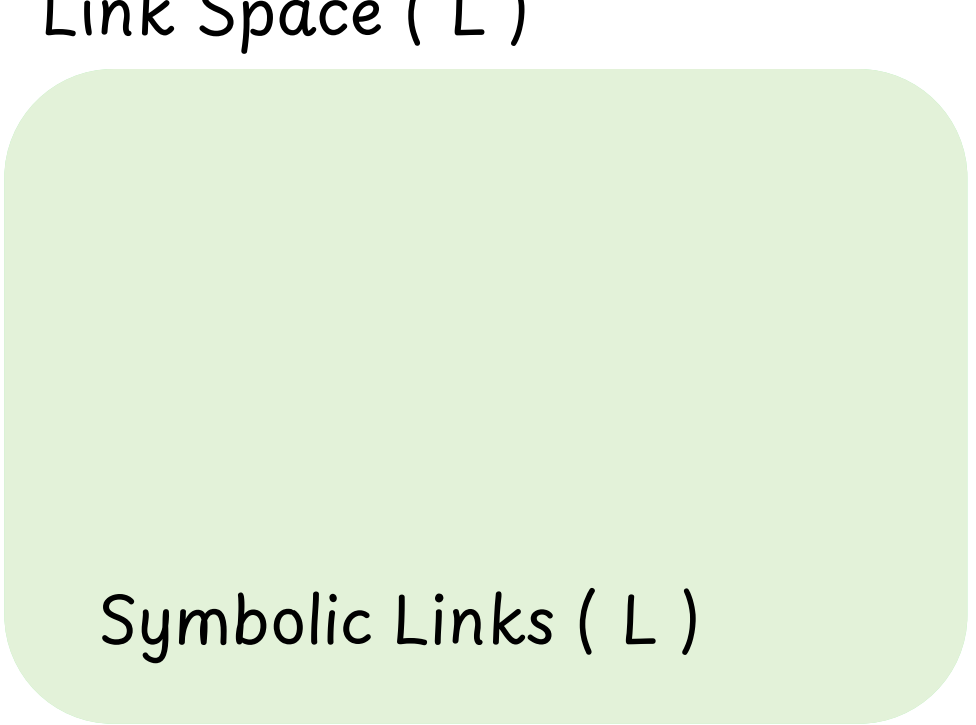
- Recall the basic method

$$N \wedge \neg P \wedge K$$

Network Property Tolerance Value

$$K = \Sigma_{l \in L} f_l \leq k$$

Encode all links



Basic Ideas - Reducing Links Space

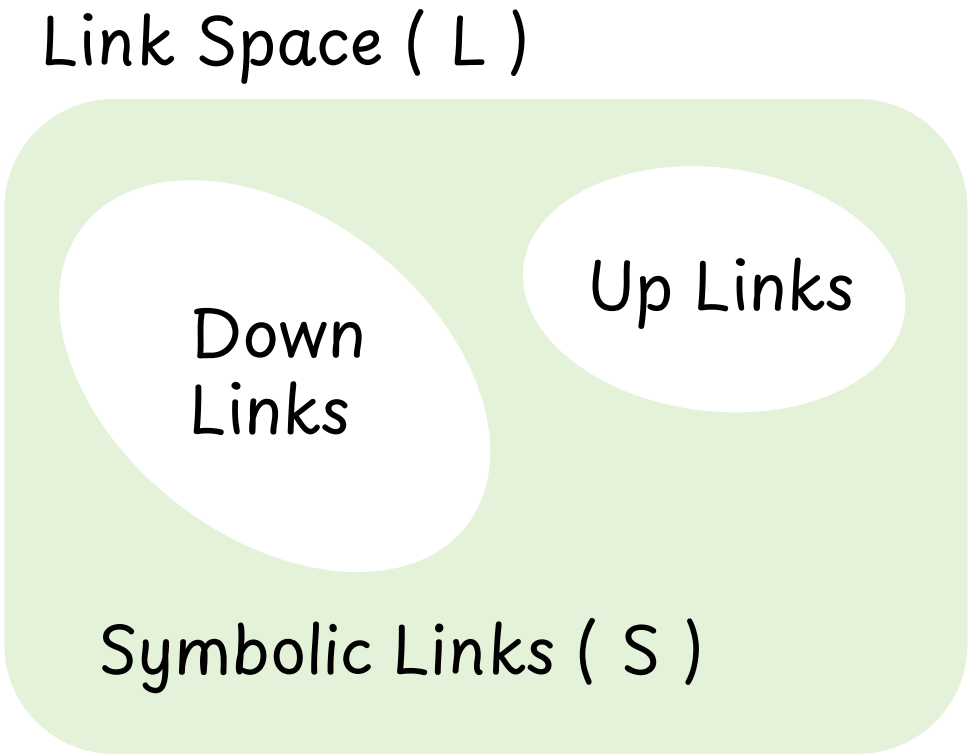
- Suppose we can decide some links status before verification

$$N \wedge \neg P \wedge K$$

Network Property Tolerance Value

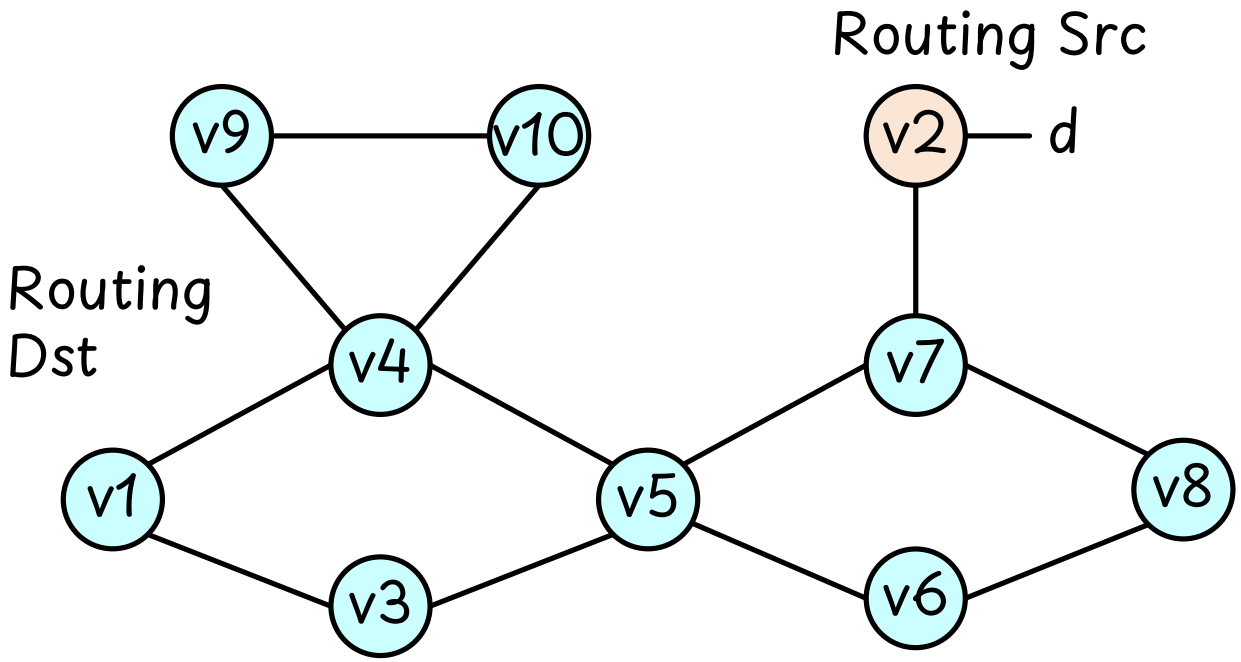
$$K = \Sigma_{l \in S} f_l \leq k$$

Encode all links



Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)



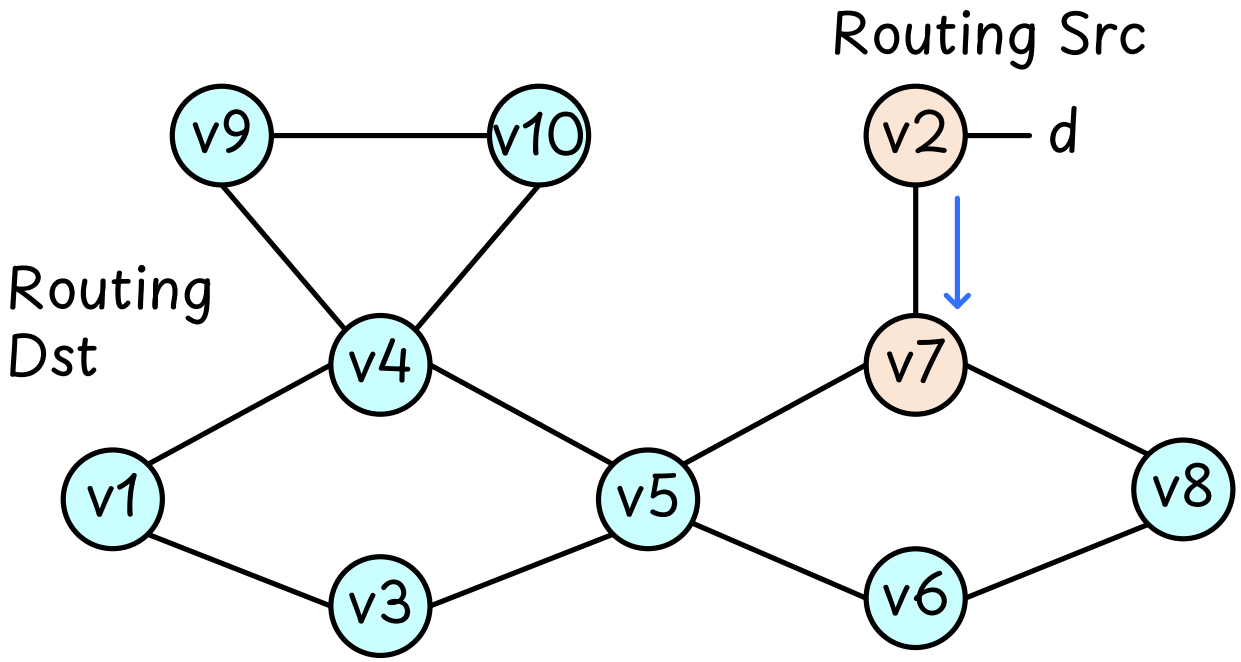
V2 Routing Table

Dst	Next_Hop	AS_Path
d	Connected	v2

Routing and forwarding directions are opposite

Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)



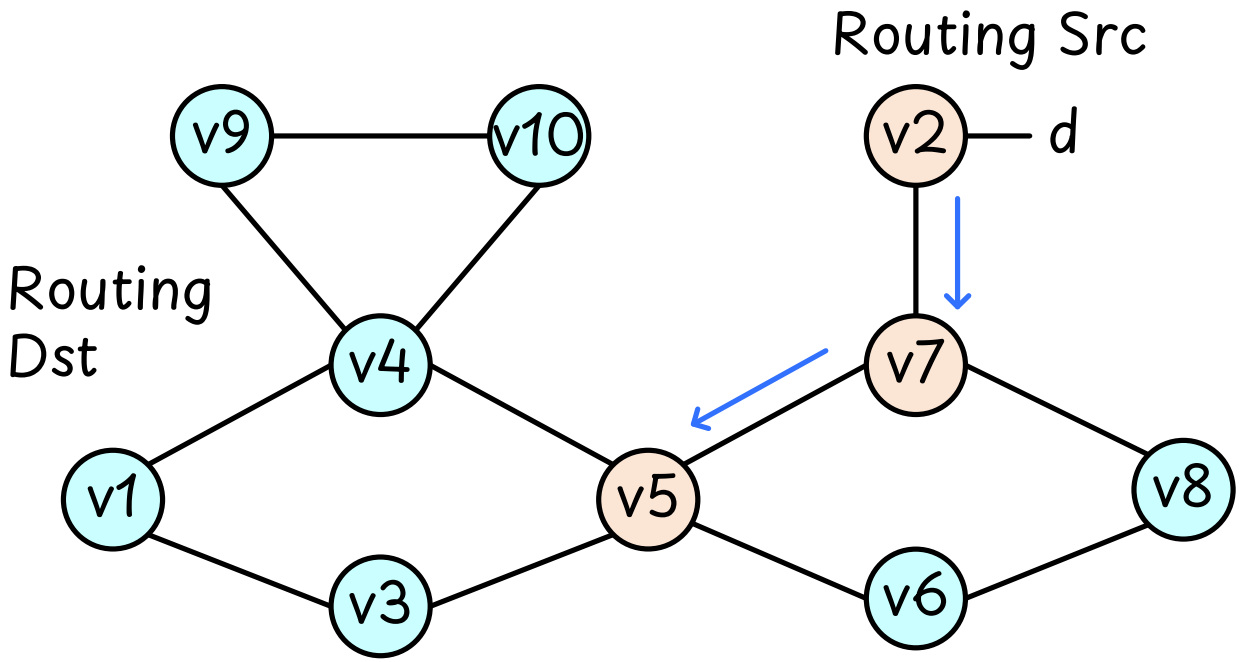
V7 Routing Table

Dst	Next_Hop	AS_Path
d	v7	v2-v7

Best route will spread to next hop

Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)



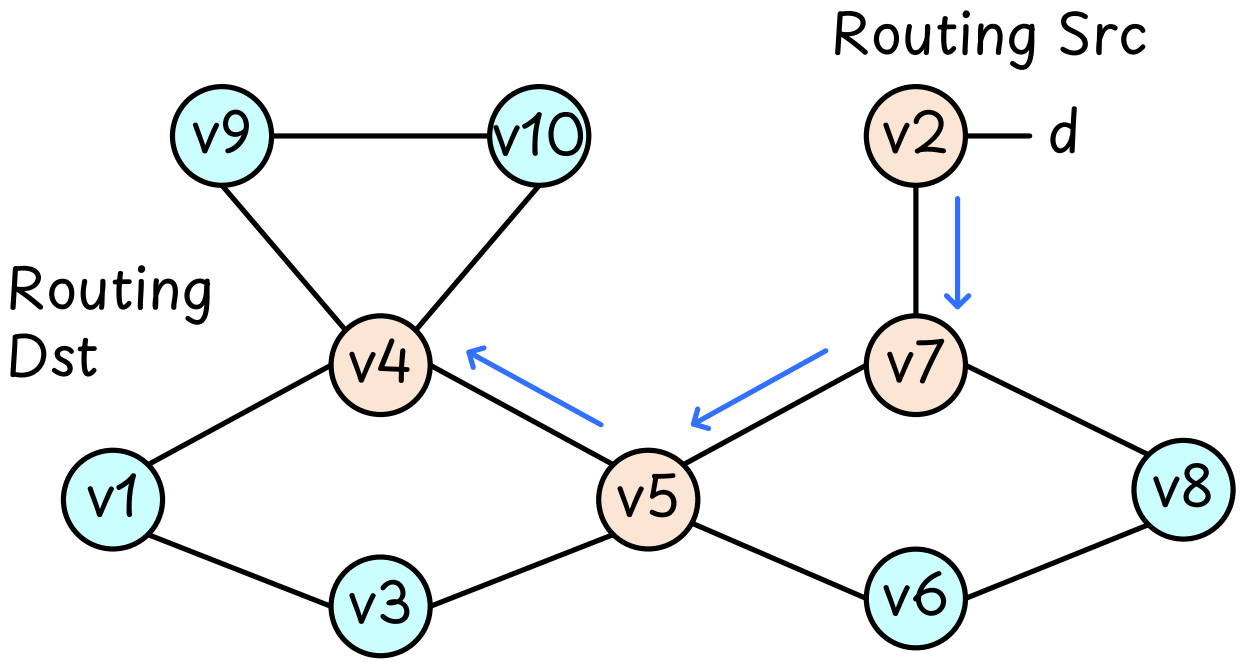
V5 Routing Table

Dst	Next_Hop	AS_Path
d	v7	v2-v7-v5

Best route will spread to next hop

Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)



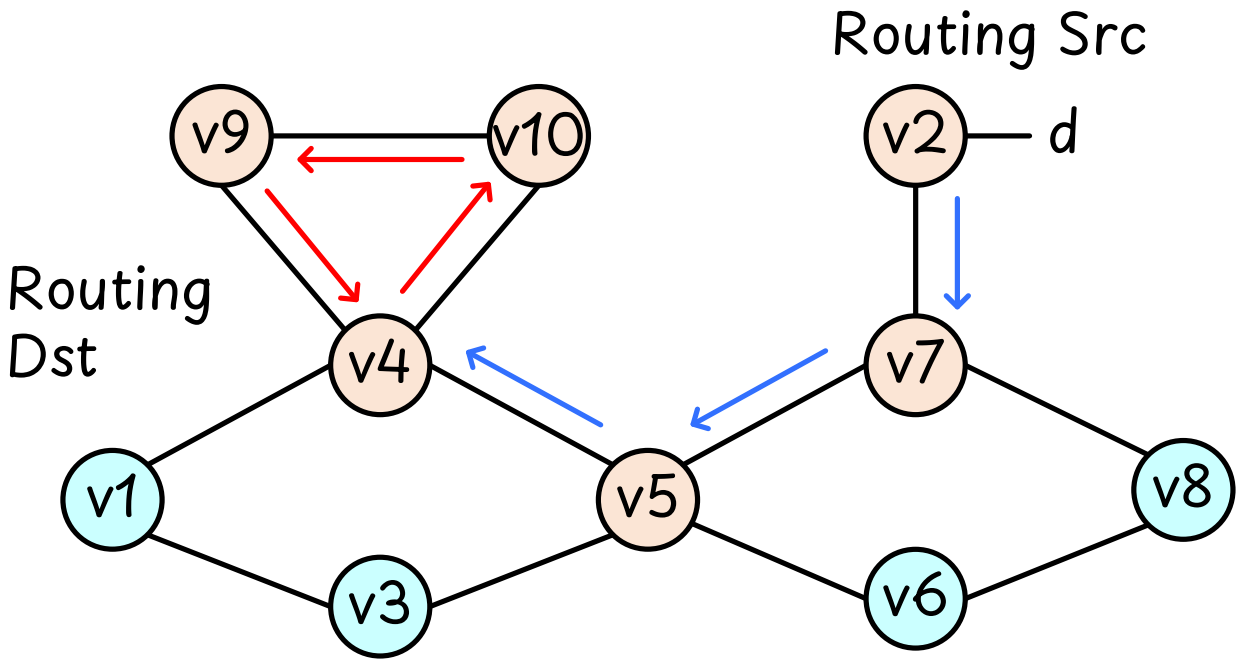
V4 Routing Table

Dst	Next_Hop	AS_Path
d	v5	v2-v7-v5-v4

Best route will spread to next hop

Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)



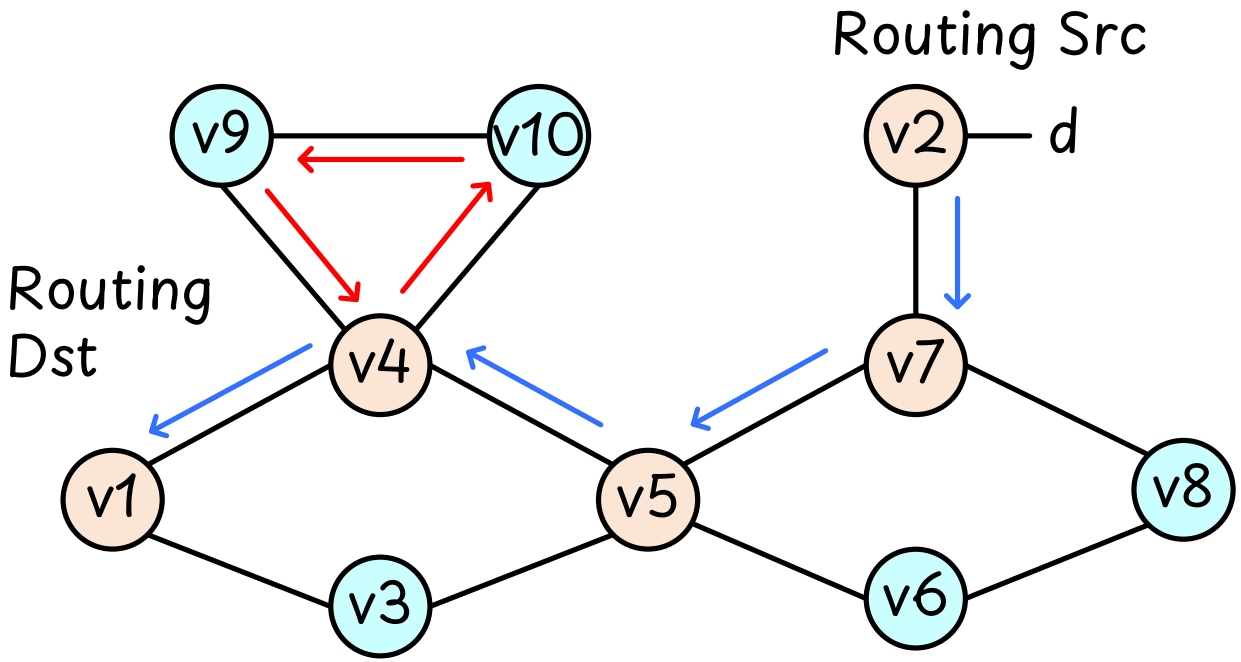
V4 Routing Table

Dst	Next_Hop	AS_Path
d	v5	v2-v7-v5-v4
~~d~~	~~v9~~	~~v2 v7 v5 v4 v10 v9 v4~~

- Routes arrive at v4 again
- The second routes will be deleted

Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)



V4 Routing Table

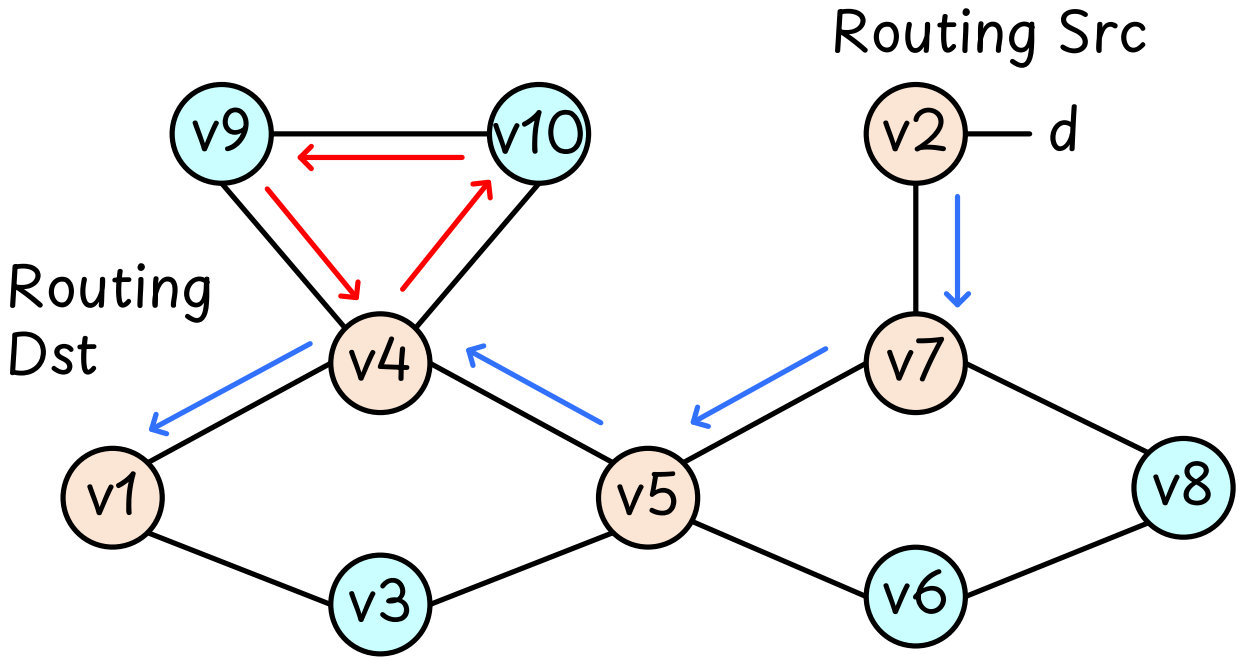
Dst	Next_Hop	AS_Path
d	v5	v2-v7-v5-v4
~~d~~	~~v9~~	~~v2-v7-v5-v4-v10-v9-v4~~

V1 Routing Table

Dst	Next_Hop	AS_Path
d	v4	v2-v7-v5-v4-v1

Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)

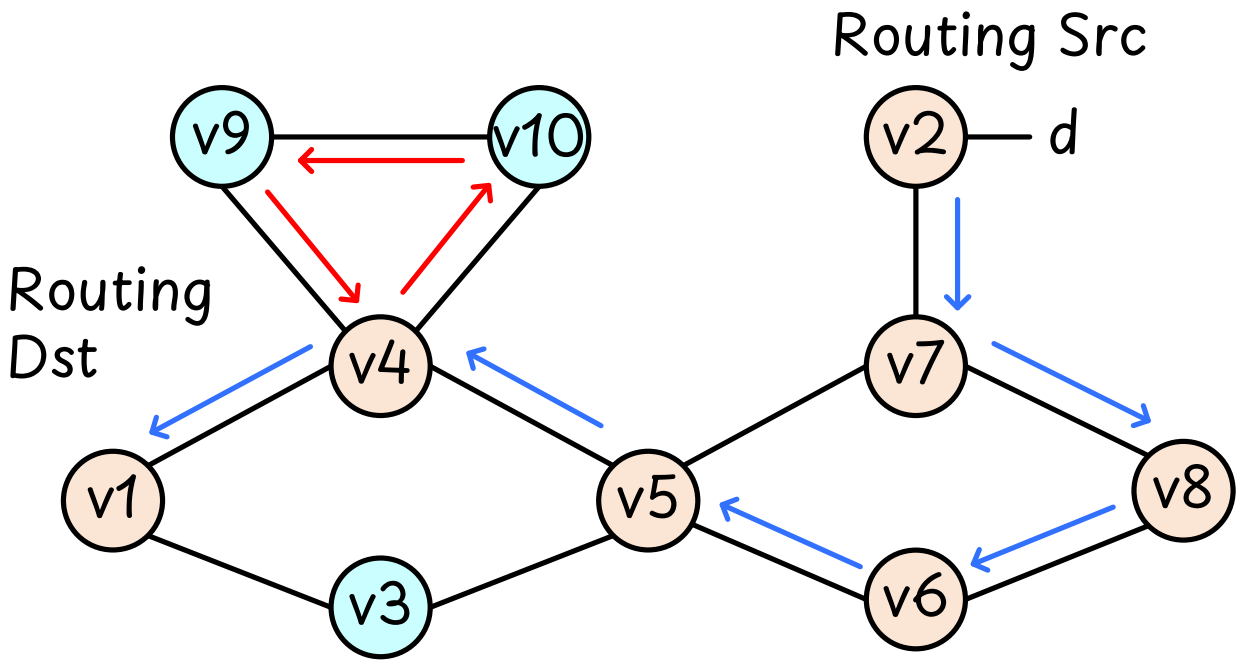


links v9-v10, v4-v10, and v4-v9

> don't spread the best routes

Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)



Even routes spread from other paths

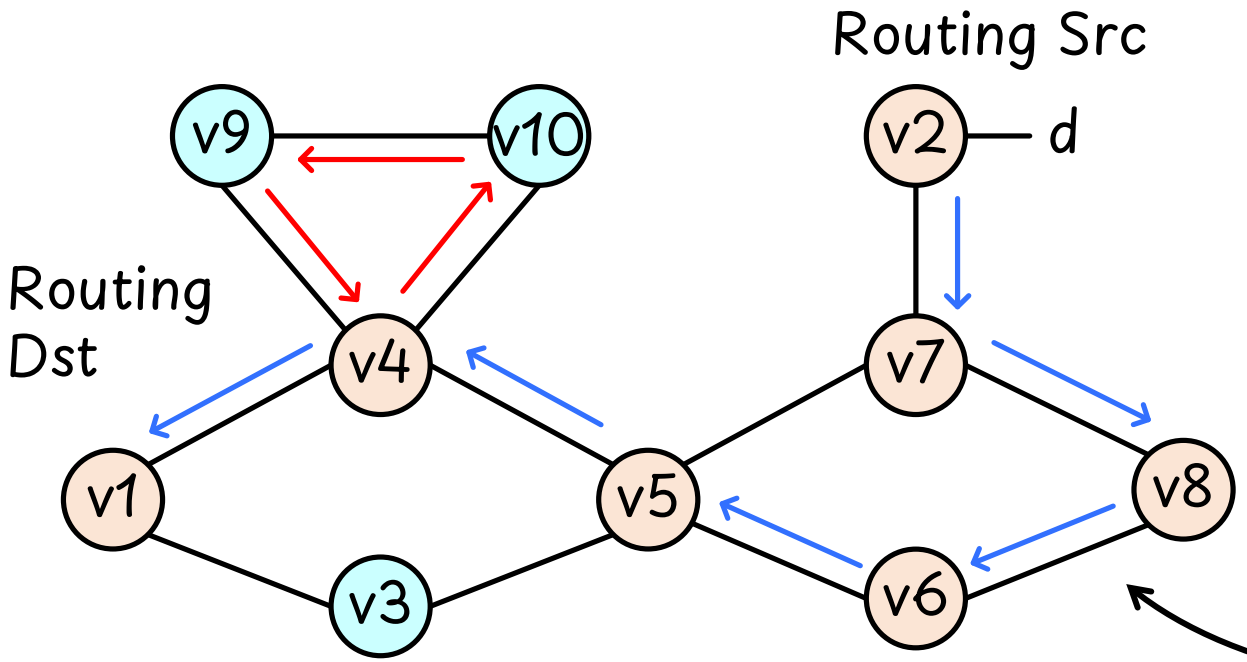
links v9-v10, v4-v10, and v4-v9

> don't spread the best routes

> can be set to "**Down**" status before verification

Basic Ideas - Down ("Irrelevant") Links

- Routing paths for Reachability(v1,v2,d,k)



We summarize :

Theorem1. if a link does not appear on any **simple path** then the link is **irrelevant**, and can be set as "**down**"

[See paper for rigorous definition]

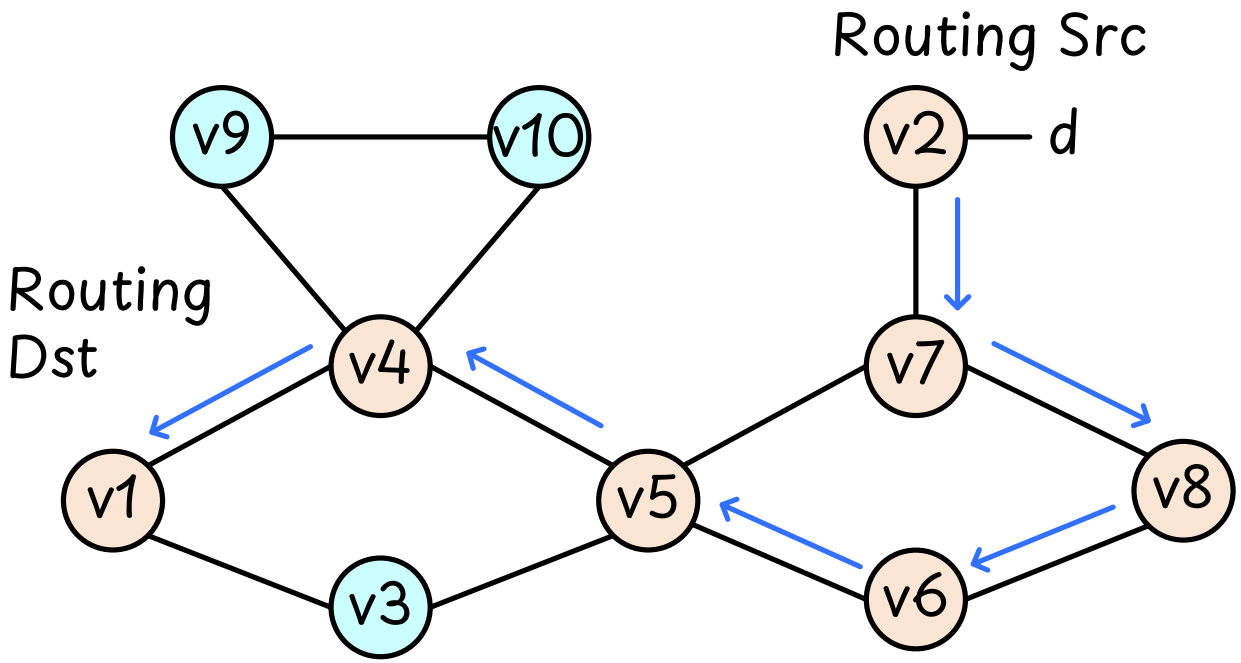
Simple path

Simple path is a loop-free path

[Graph Theory 2024]

Basic Ideas - Up ("Equivalent") Links

- Routing paths for Reachability(v1,v2,d,k)

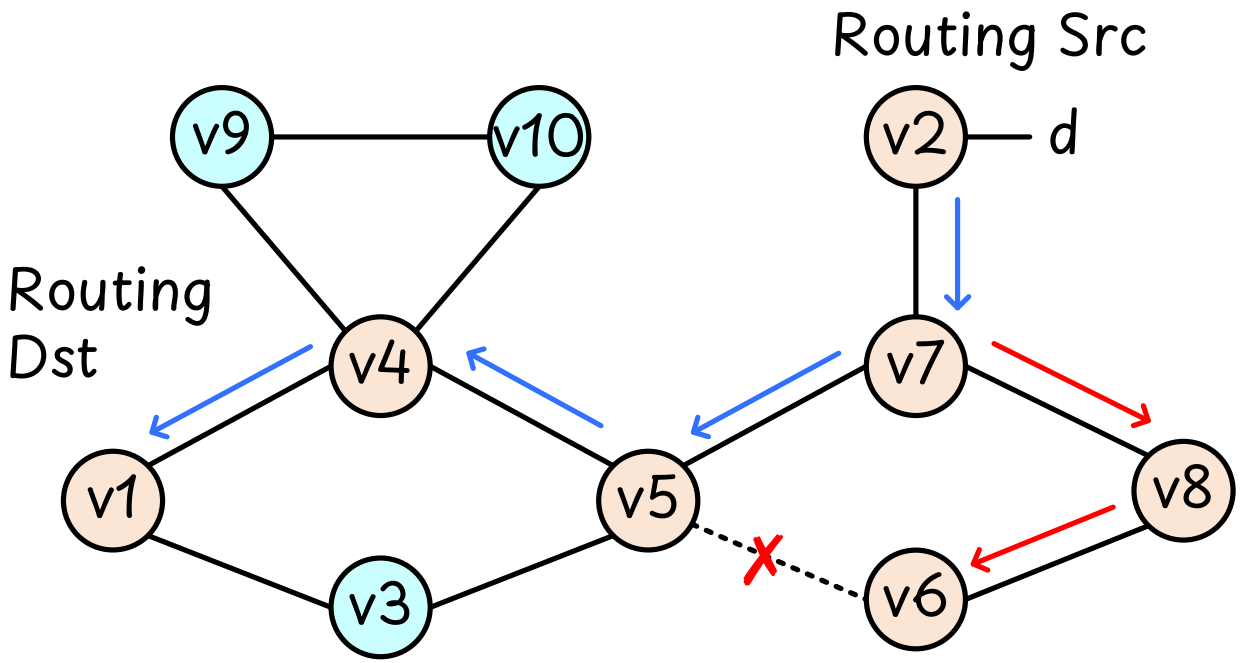


Suppose the best routes spread via :

v2-v7-v8-v6-v5-v4-v1

Basic Ideas - Up ("Equivalent") Links

- Routing paths for Reachability(v1,v2,d,k)



Suppose the best routes spread via :

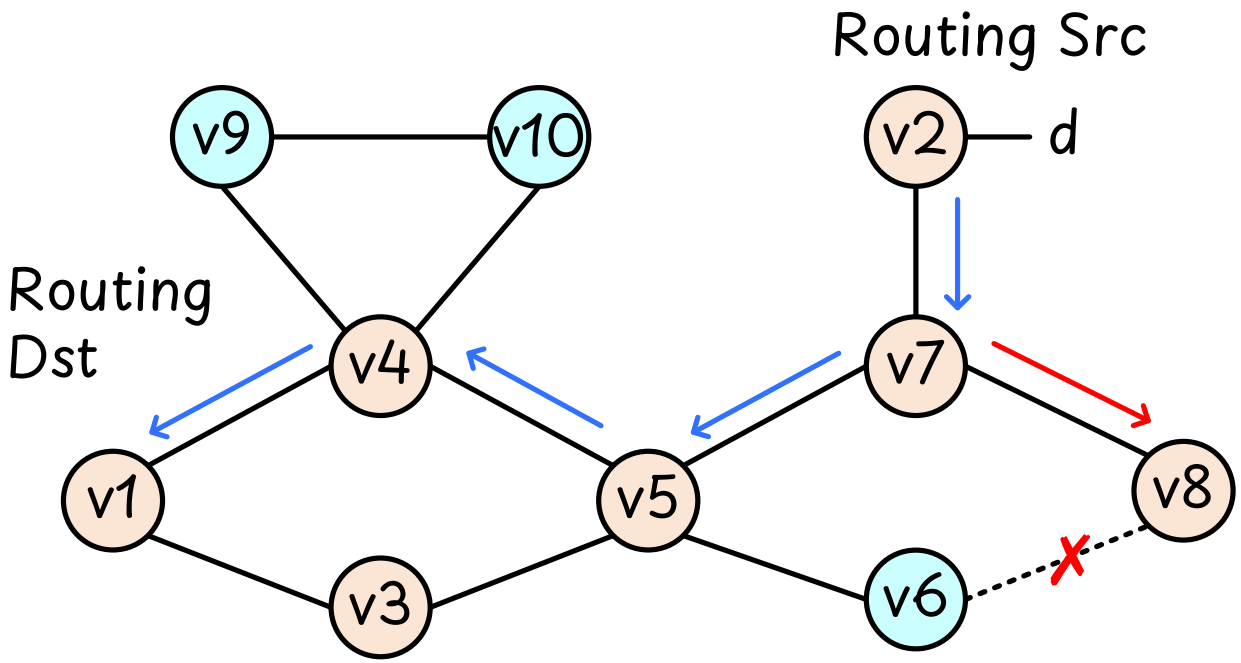
v2-v7-v8-v6-v5-v4-v1

> Failure scenario 1 (Failing v5-v6):

v2-v7-v5-v4-v1 (Routing path)

Basic Ideas - Up ("Equivalent") Links

- Routing paths for Reachability(v1,v2,d,k)



Suppose the best routes spread via :

v2-v7-v8-v6-v5-v4-v1

> Failure scenario 1 (Failing v5-v6):

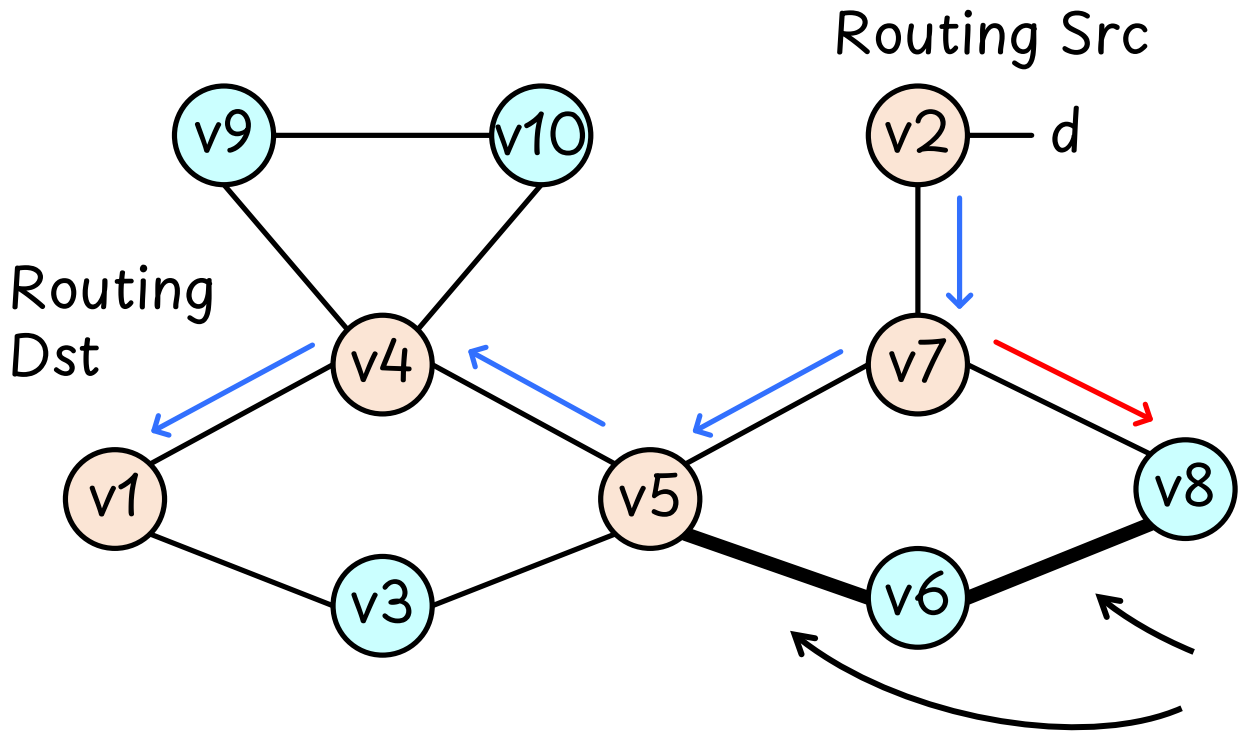
v2-v7-v5-v4-v1 (Routing path)

> Failure scenario 2 (Failing v6-v8):

v2-v7-v5-v4-v1 (Routing path)

Basic Ideas - Up ("Equivalent") Links

- Routing paths for Reachability(v1,v2,d,k)



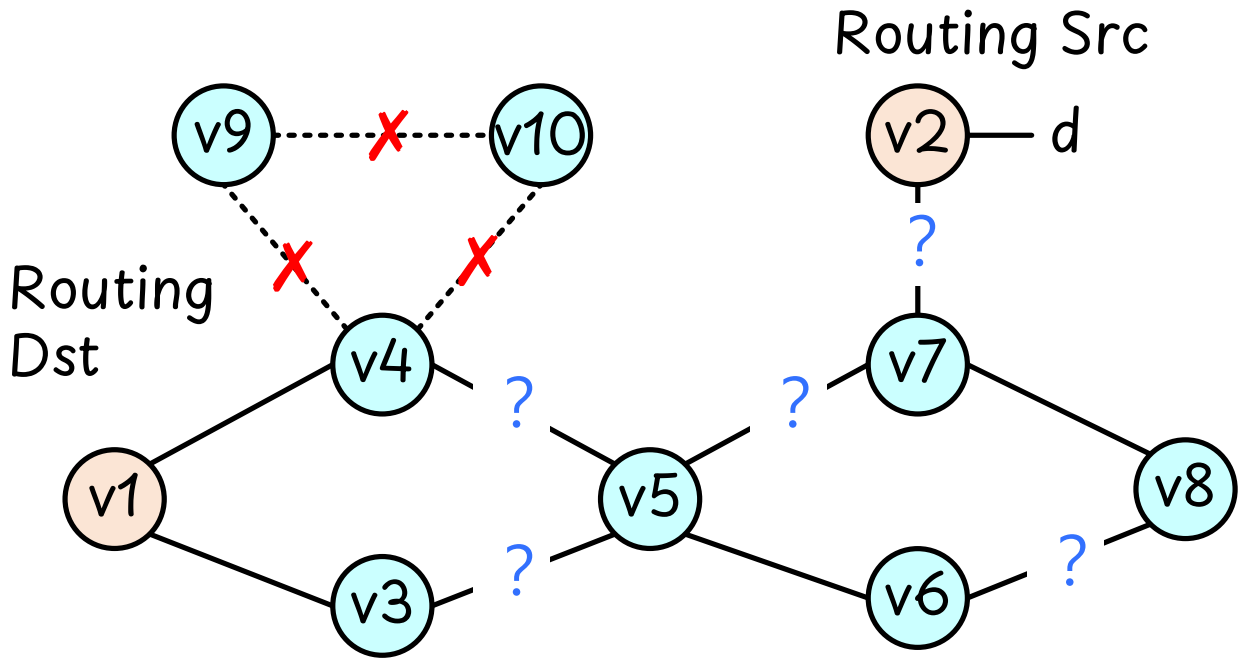
We summarize :

Some links is **equivalent**, and one of them can be set as **"Up"**

[See paper for rigorous definition]

Basic Ideas

- Routing paths for Reachability(v1,v2,d,k)



- Before reducing space

  Symbolic links : 12

  Failure scenarios : C(12, 3) = 220

- After reducing space

  Symbolic links : 5

  Failure scenarios : C(5, 3) = 10

Find irrelevant and equivalent links is challenging

- Enumerating all simple paths is NP-Hard Problem*

  It need more than 4h, but verify a property needs only 0.5h

- Checking equivalent links is chicken-and-egg problem

  We want to speedup verification, but it needs verification again

* A new algebraic approach to finding all simple paths and cycles in undirected graphs [ICIA'15]

We propose

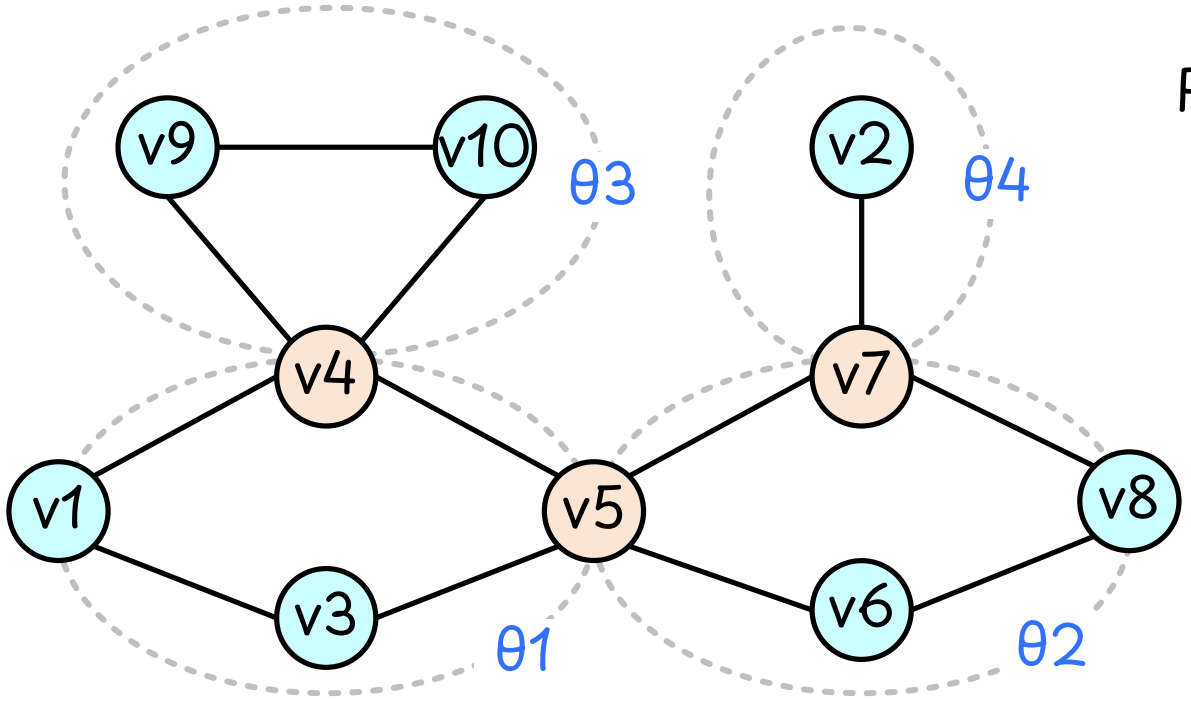
Link Pruning and Link Compression

Link Pruning - Target

Find all simple paths without enumerating all paths

Step1 Getting Point biconnected component

PBC : a subgraph that removing any single link still preserves connectivity between every pair of nodes



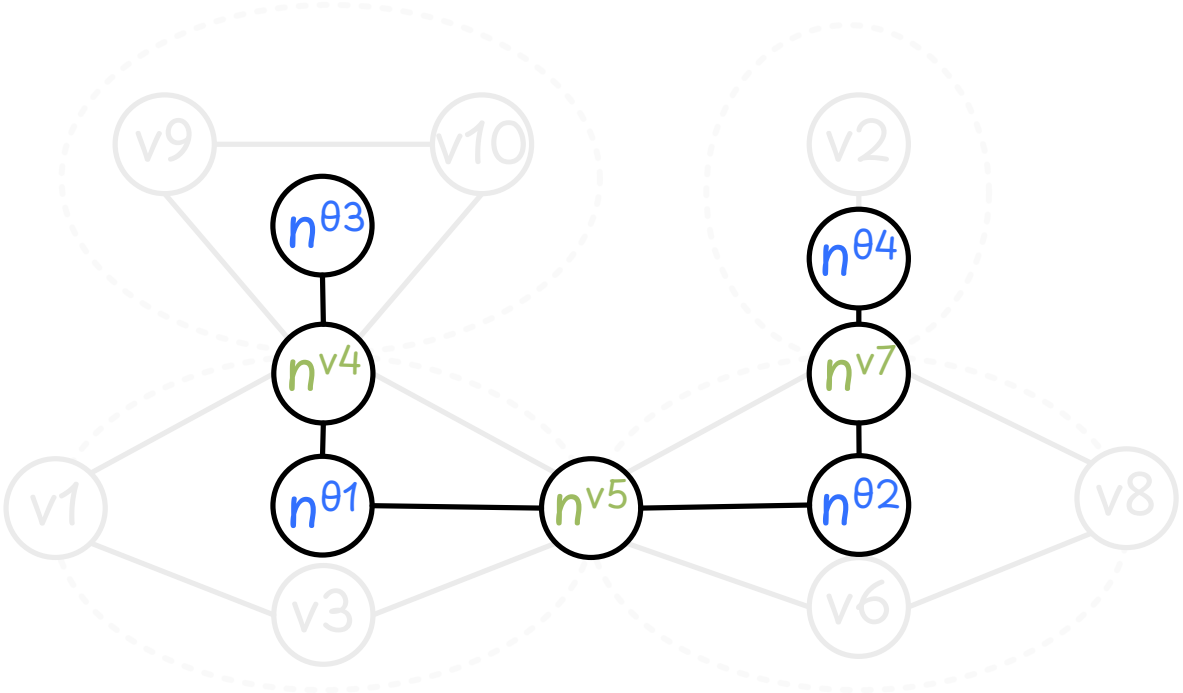
Point biconnected component $\theta 1=\{v1,v3,v4,v5\}$

Point biconnected components $\Theta=\{\theta 1,\theta 2,\theta 3,\theta 4\}$

Cut points $\eta=\{v4,v5,v7\}$

Step2 - Building Block Cut Tree

Use cut points and components to build a graph T

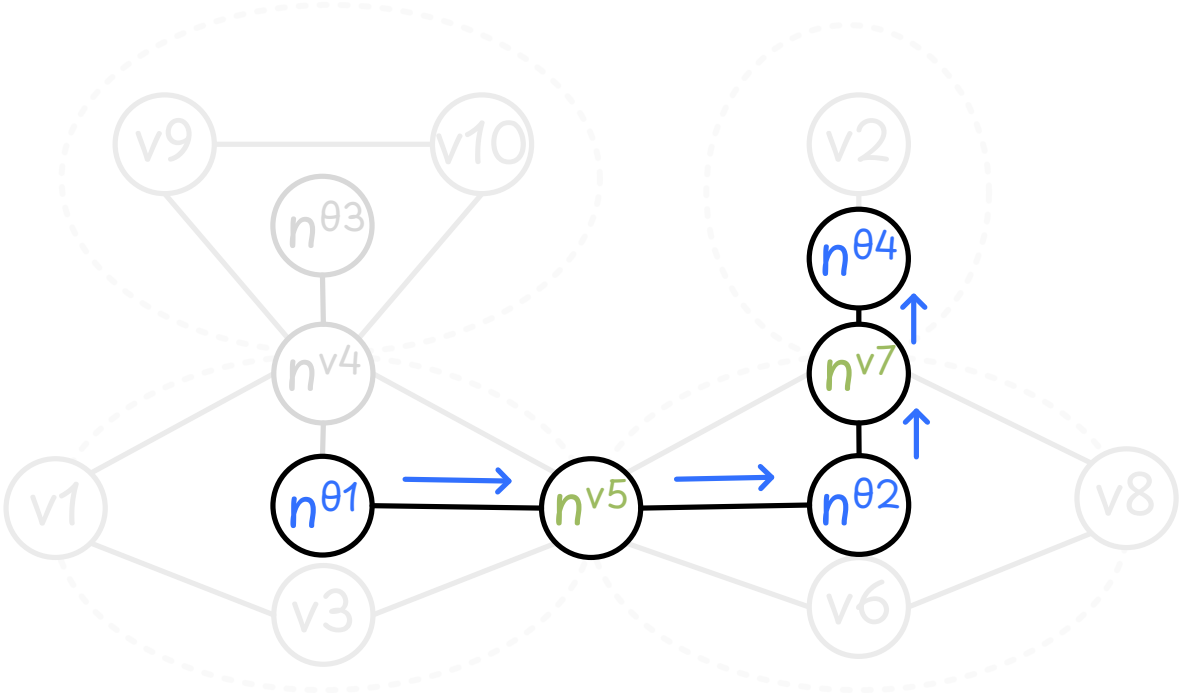


$N^{\theta} = \{n^{\theta} | \theta \in \Theta\}$, $N^{v} = \{n^{v} | v \in \eta\}$

$E = \{e | e = (n^{v}, n^{\theta}) \vee (n^{\theta}, n^{v}), v \in \theta\}$

$T = (N^{v} \cup N^{\theta}, E)$

Step3 - Traversing the block-cut tree

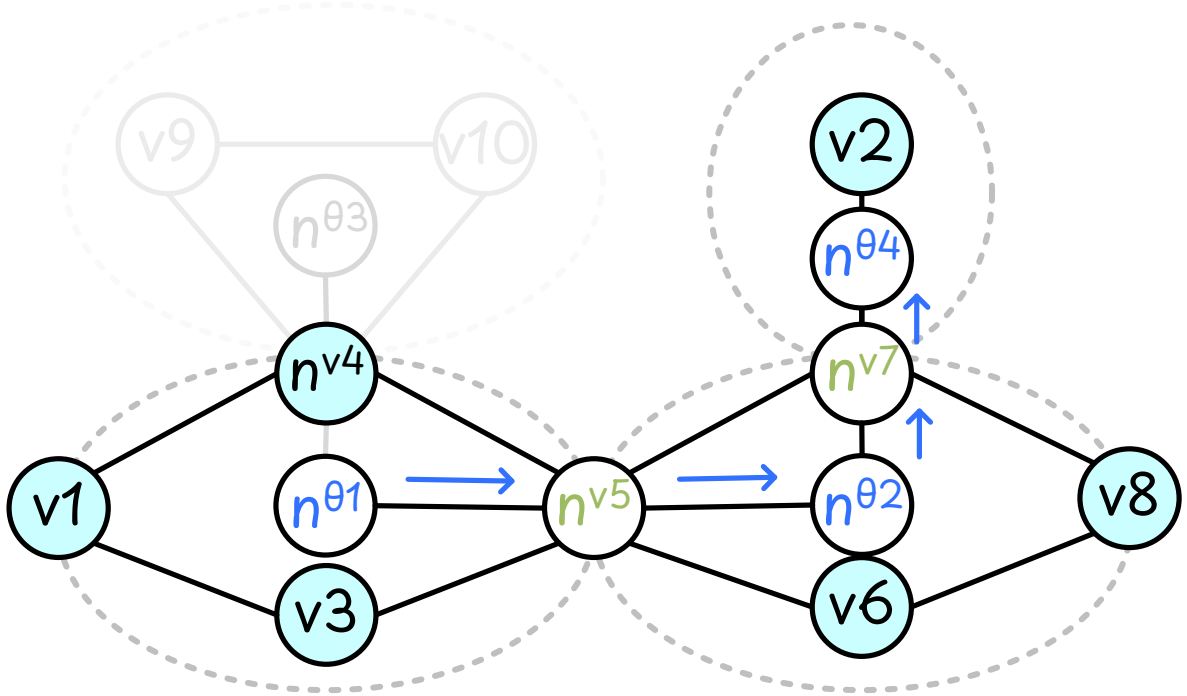


Source node : $n^{\theta 1}$

Destination node : $n^{\theta 4}$

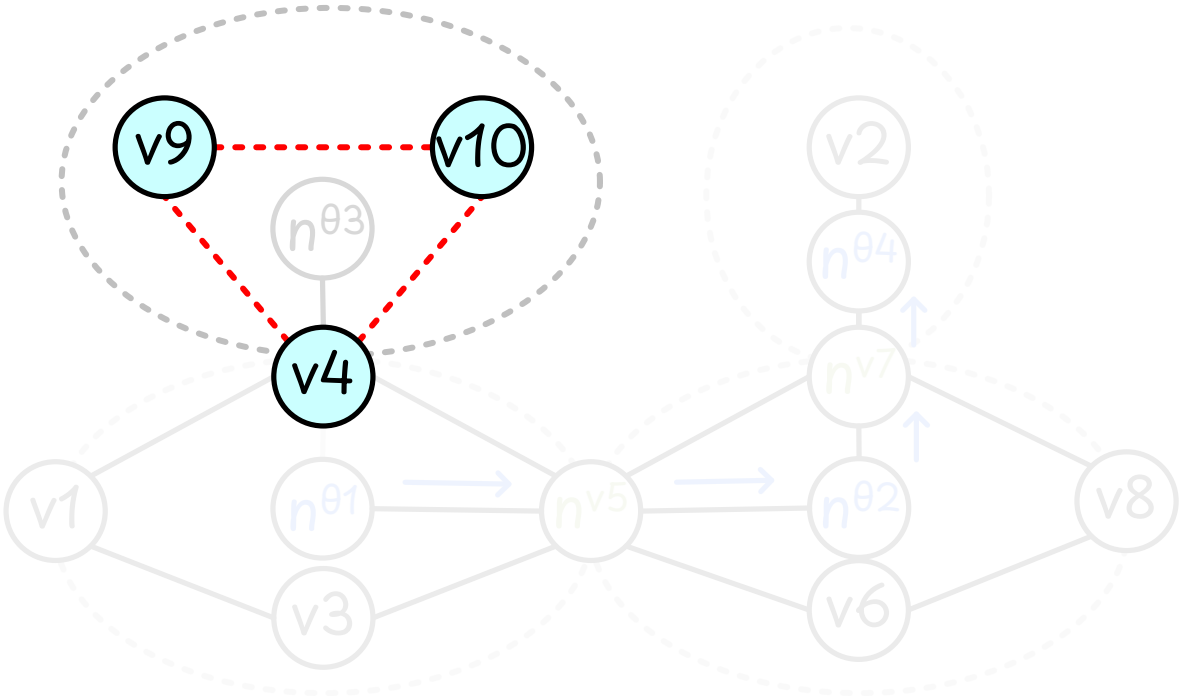
Step3 - Calculating links in simple paths

Combine all links in traversed components as L'



Step3 - Calculating the irrelevant links

Return the complement set of L' as irrelevant links



v9-v10, v4-v9, v4-v10 are **"irrelevant"** links

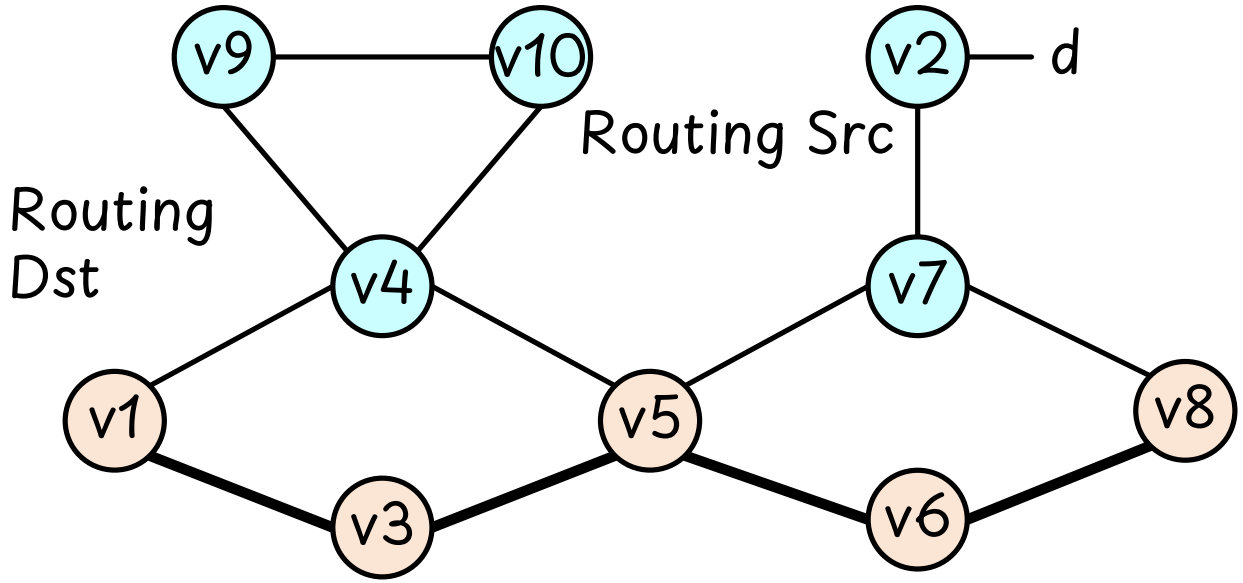
[We propose Theorem 2 to provide correctness, see our paper for details]

Link Compression - Target

Instead of identifying all equivalent links, we try to find a subset of equivalent links

Prerequisite - Trivial Path

If all nodes within the paths (expect end points), their degree is 2, this path is trivial path.

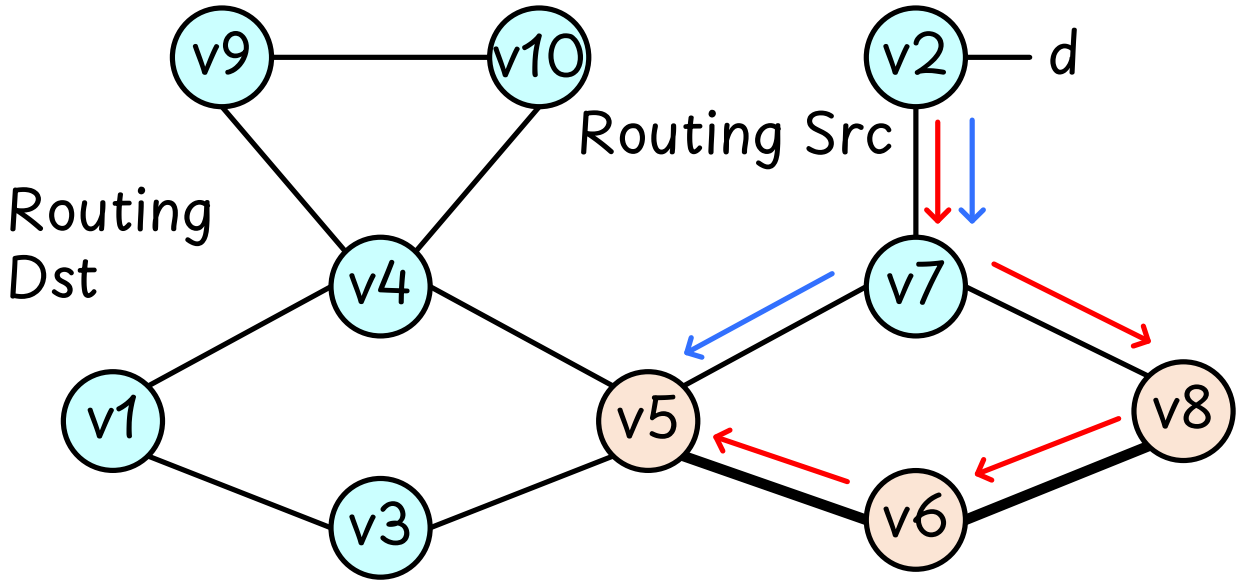


Trivial Path1 :
v7-v8-v6-v5

Trivial Path2:
v5-v3-v1

Link Compression

If the degree of nodes is 2, then adjacent links are equivalent

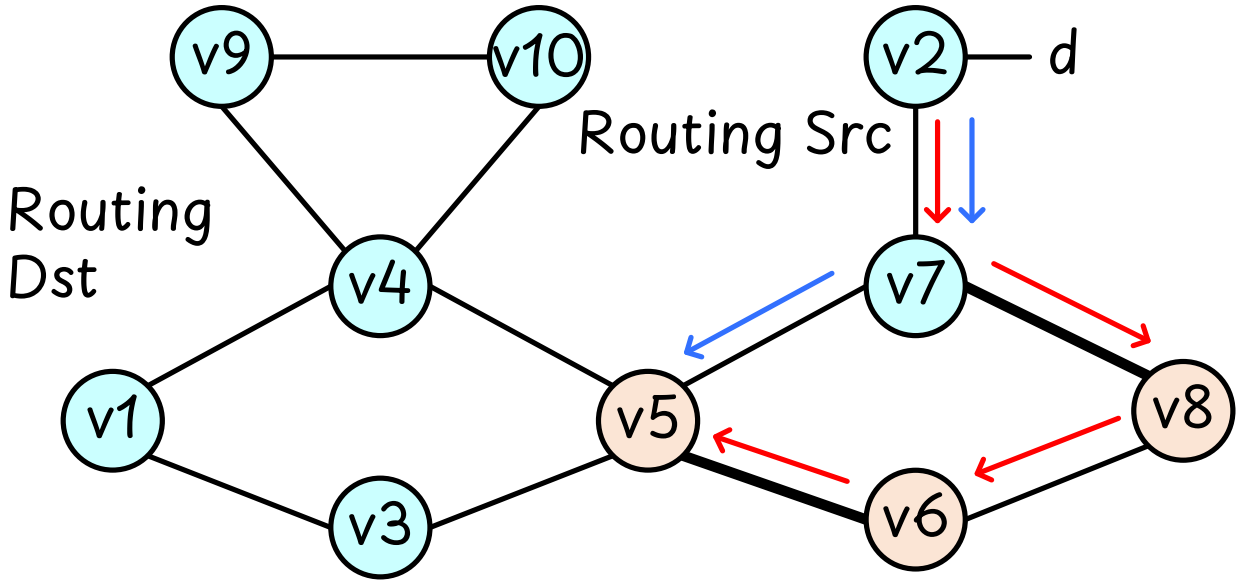


- v5-v6, v6-v8 are "equivalent"
- v6-v8, v7-v8 are "equivalent"

Reason: Failing both of them blocks the route from choosing this subpath.

Link Compression

If the degree of nodes is 2, then adjacent links are equivalent



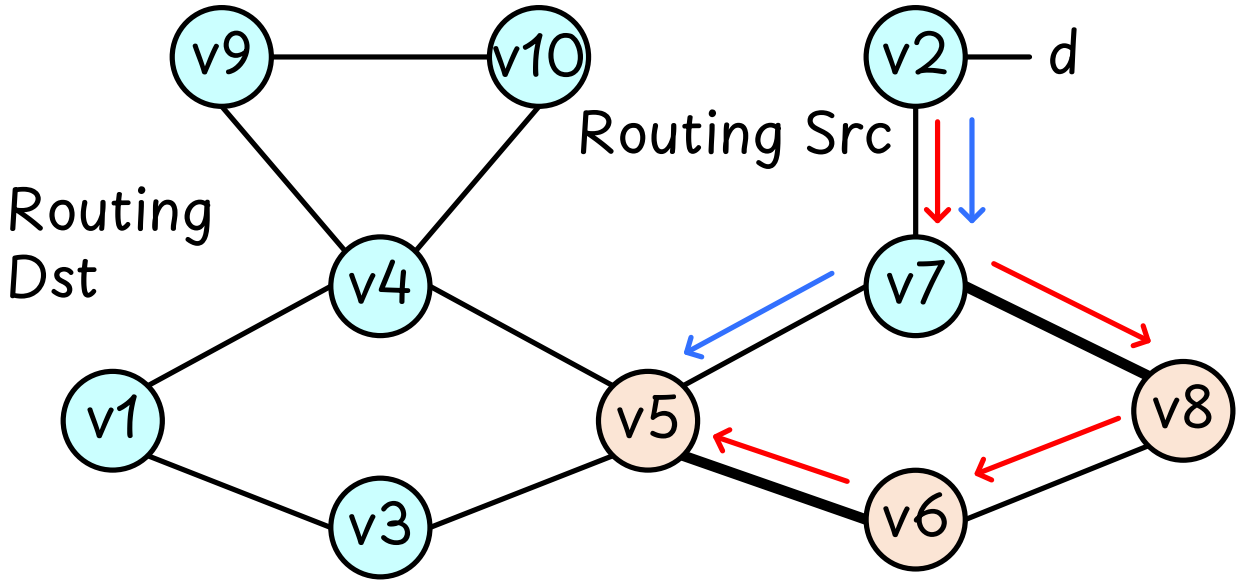
- v5-v6, v6-v8 are "equivalent"
- v6-v8, v7-v8 are "equivalent"

Equivalence is transitive

- v5-v6, v7-v8 are "equivalent"

Link Compression

The links in trivial path is equivalent



We summarize :

> Theorem 3. The links in trivial path is equivalent

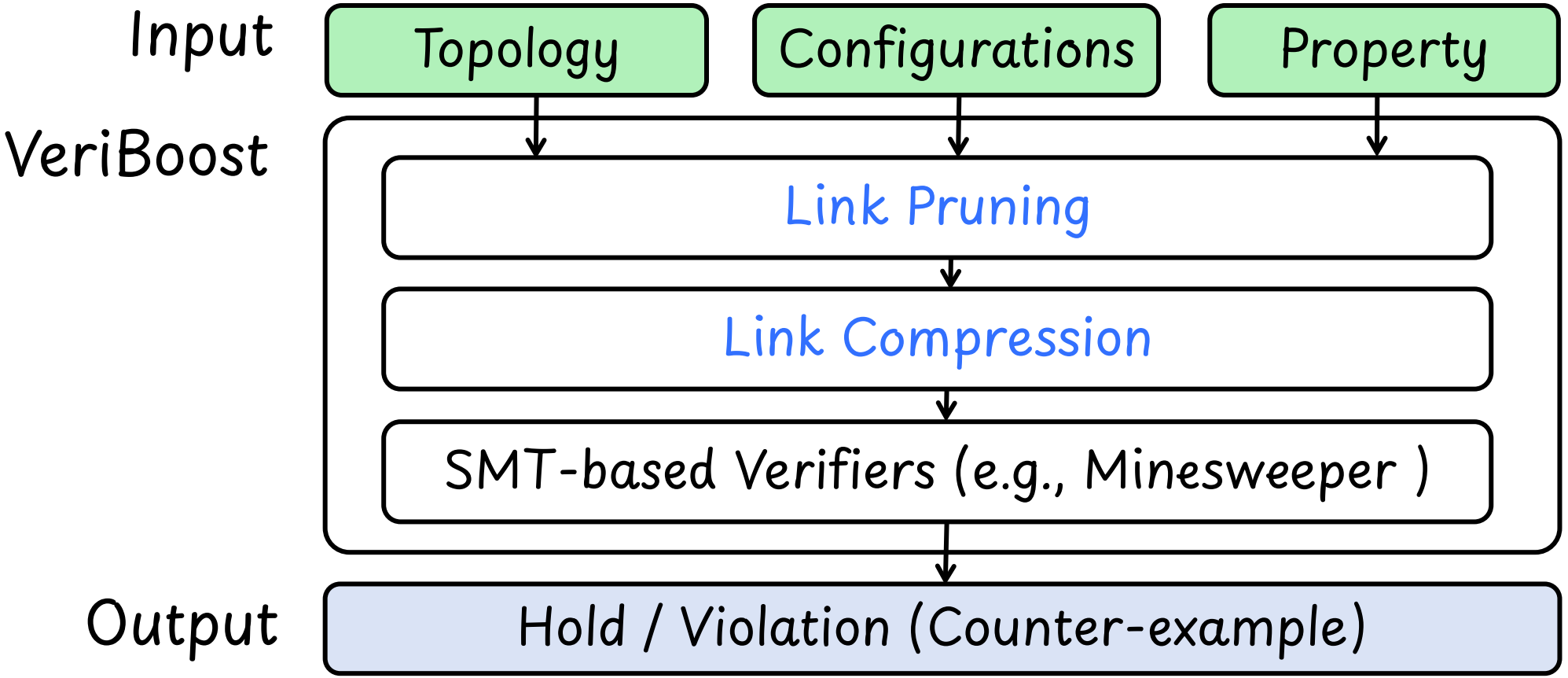
[see our paper for details]

Optimization

Property Trimming

- If the tolerance value of a property to be verified is k, there must be more than k disjoint paths in the topology.

Implementaion



4k~ lines of Java code, open source on github.com/XJTU-NetVerify/Veriboost

Evaluation Setup - Datasets

- 260 real WAN topologies from https://topology-zoo.org/
- 9 configurations from NetSMT TON'22 Config2Spec NSDI'22

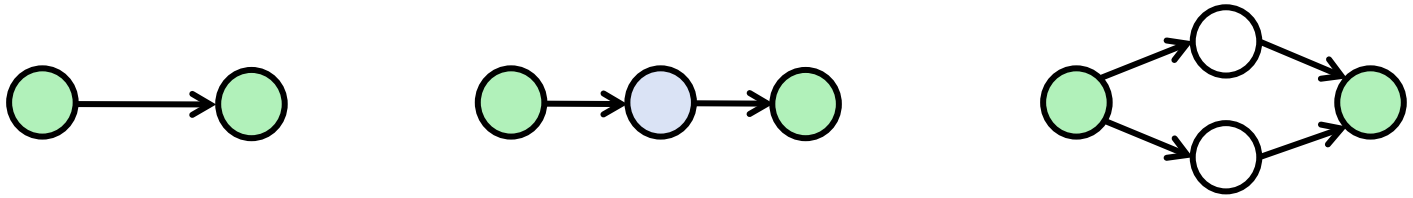
Datasets	Ren	Arn	Bic-O	Bic	Esn	Lat	Col-O	Col	Clt	Usc-O	Usc	Cog
Category	Small				Medium				Large			
Nodes	34	35	33	33	69	70	70	70	154	158	158	198
Links	44	47	48	48	80	75	85	85	178	189	189	244
Lines ($\times 10^3$)	4.8	5.0	3.0	6.9	9.2	9.1	6.1	11.5	20.6	13.7	22.0	27.1

"-O" means OSPF configurations, others are BGP configurations

Evaluation Setup - Properties

Total number of properties $3 \times 25 \times 4 = 300$

- Type Reachability Waypointing Load balancing

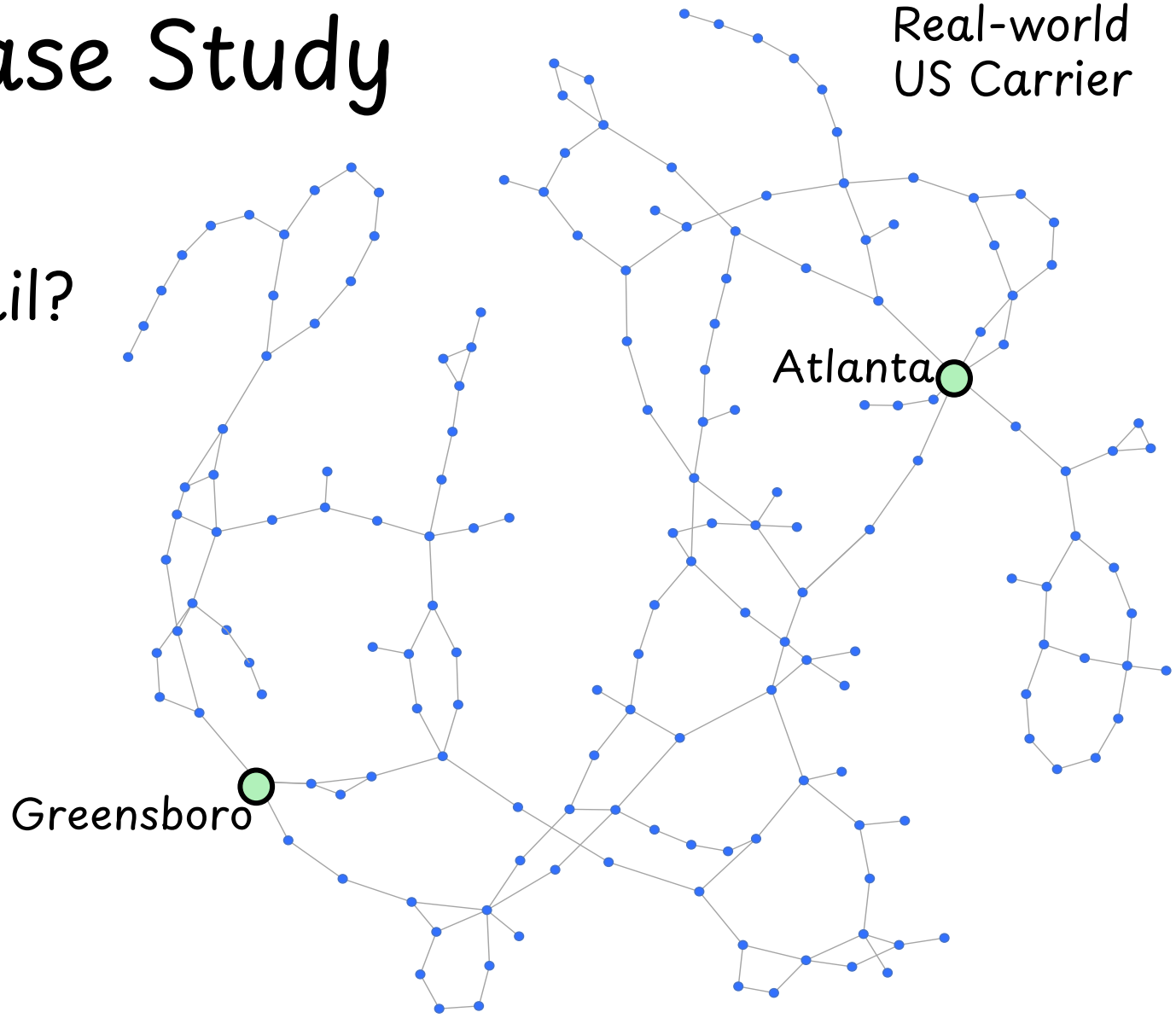


- Tolerance Value $k = 0, 1, 2, 3$

- Number of pairs 25

Effectiveness - Case Study

Can Greensboro still reach Atlanta if any three links fail?



Effectiveness - Case Study

Can Greensboro still reach Atlanta if any three links fail?

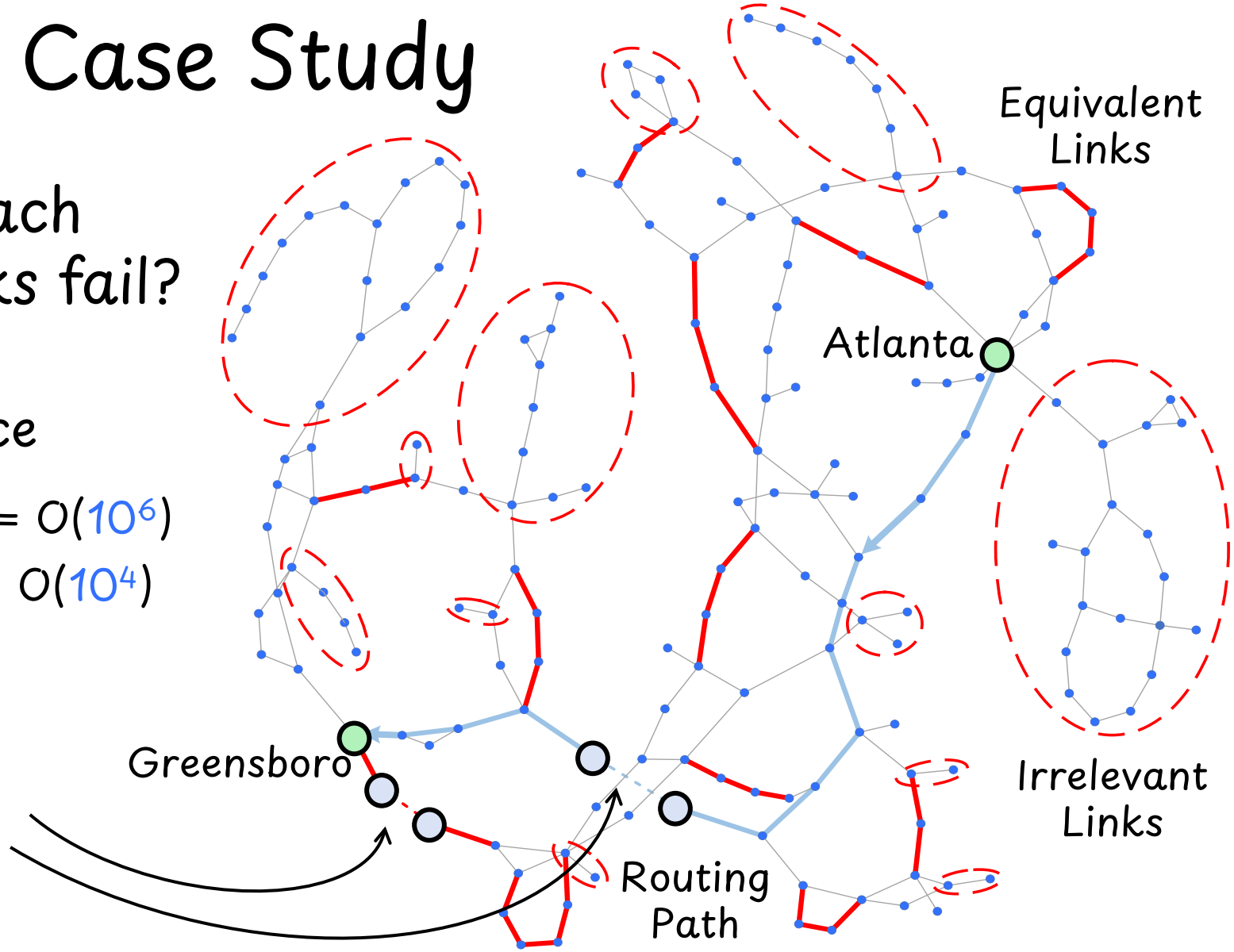
- The failure scenario space

  Without VeriBoost C(189,3) = O(10^6)

  With VeriBoost C(61,3) = O(10^4)

- The property is violation

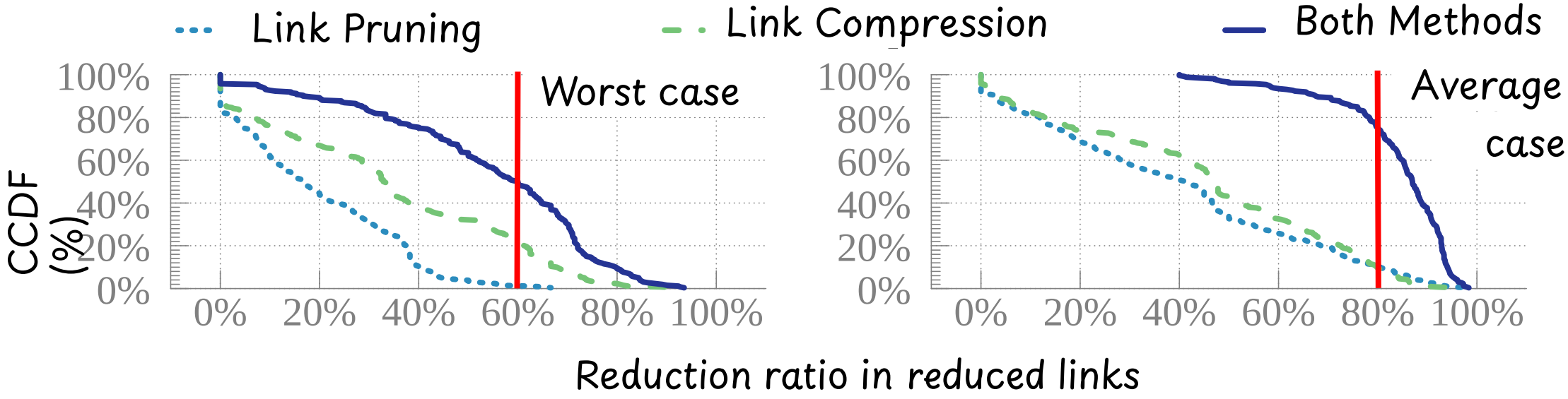
  "Salisbury–Concord" and "Burlington–Durham" fail



Effectiveness

Experiment across 260 real WAN topologies

- VeriBoost can reduce 50% of links for 60% of topologies (Worst case).
- VeriBoost can reduce 78% of links for 80% of topologies (Average case).



Correctness

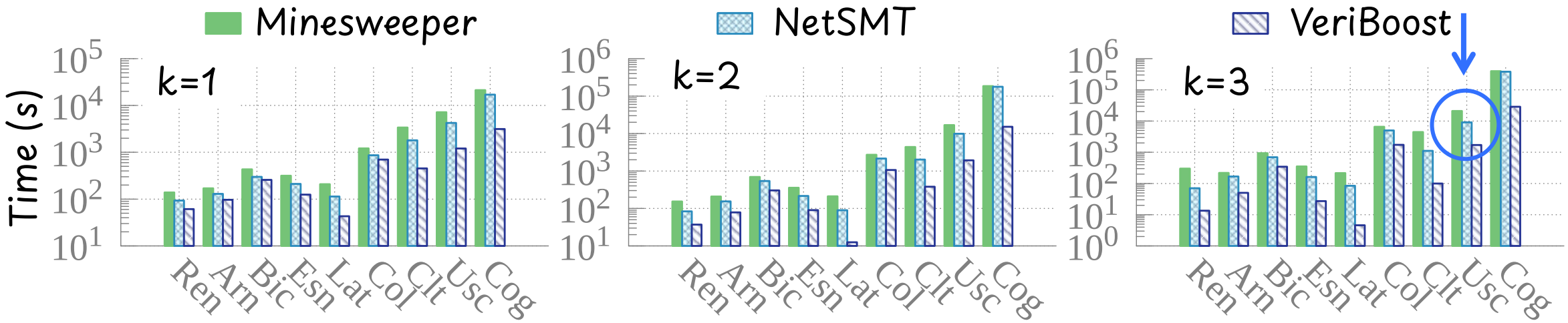
The properties verified by Minesweeper, NetSMT, and VeriBoost are identical

Network	BGP									OSPF		
	Ren	Arn	Bic	Esn	Lat	Col	Clt	Usc	Cog	Bic	Col	Usc
k=1	94	150	150	112	62	140	100	112	150	151	140	112
k=2	44	100	100	62	12	90	50	62	100	100	90	62
k=3	12	50	50	50	0	40	0	12	50	50	40	12

Speedup

Experiment across 9 networks

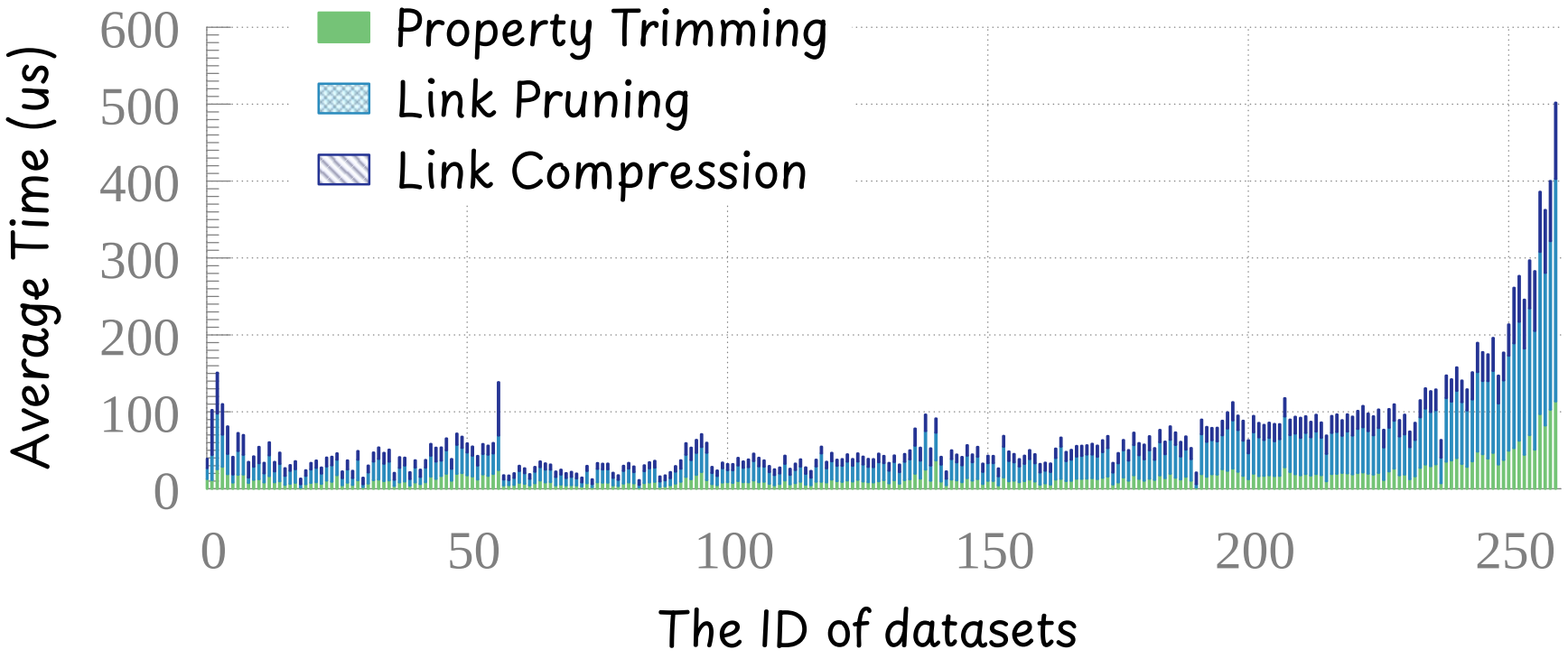
- VeriBoost speeds up Minesweeper by 2-47×
- The SOTA speeds up Minesweeper by only 2-3×



Microbenchmark

Experiment across 260 real WAN topologies

- The total preprocessing time is all within the millisecond range



Conclusion

[Background] Verifying Fault Tolerance for WANs is challenging

due to the huge failure scenario space

[SOTA] Existing speedup methods are limited

to symmetric topology structures or specific routing policies

[Method] We propose VeriBoost to improves scalability

via pruning irrelevant links and compressing equivalent links

[Experiment] VeriBoost achieves a 2-47× speedup

with millisecond-level preprocessing

西安交通大學
XI'AN JIAOTONG UNIVERSITY

Happy to take your questions

FM2026

The 27th International Symposium on Formal Methods

Tokyo, Japan
May 18 - 22, 2026