Improved Fast Rerouting Using Postprocessing

Klaus-Tycho Foerster (University of Vienna, Austria)

Andrzej Kamisiński (AGH University of Science and Technology in Kraków, Poland)

Yvonne-Anne Pignolet (DFINITY, Switzerland)

Stefan Schmid (University of Vienna, Austria)

Gilles Tredan (LAAS-CNRS, France)

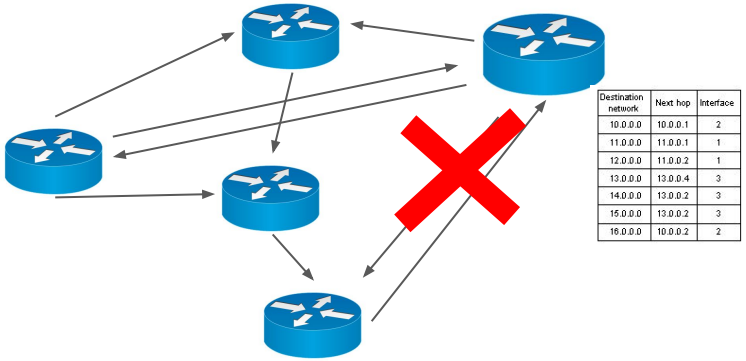
A tale of arborescences and donuts..

...and their connection to routing

Outline

1. Model and Objectives
2. Arborescence-based Fast Rerouting
3. Postprocessing Framework
4. Case Studies
5. Conclusion and Outlook

Motivation



Destination network	Next hop	Interface
10.0.0.0	10.0.0.1	2
11.0.0.0	11.0.0.1	1
12.0.0.0	11.0.0.2	1
13.0.0.0	13.0.0.4	3
14.0.0.0	13.0.0.2	3
15.0.0.0	13.0.0.2	3
16.0.0.0	10.0.0.2	2

Static Fast Rerouting (FRR)

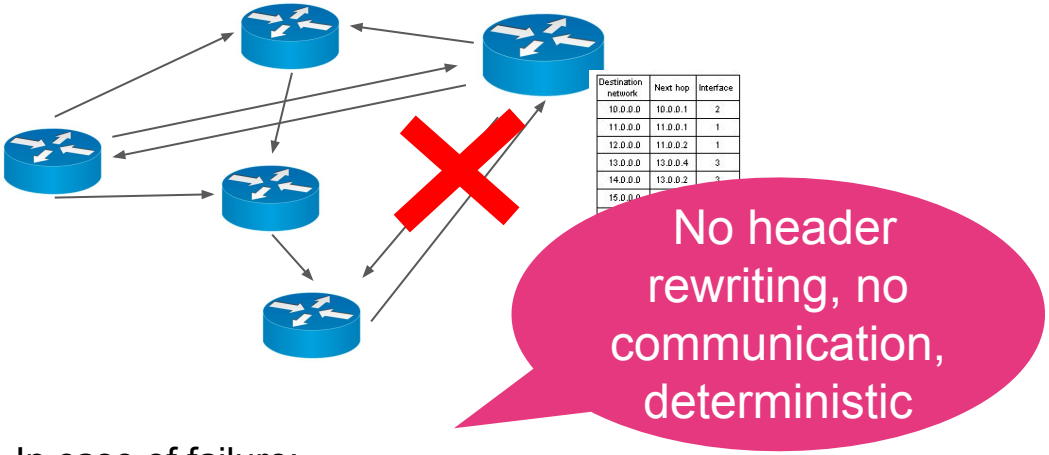
- Seamless failover
- Precomputed failover-routes

Approaches for maximal resilience are known [Chiesa et al. TON17]
=> What about stretch, load and other performance criteria?
[CCR18,Infocom19,DSN19]
=> Despite NP-hardness results and beyond special cases?

Model and Objectives

Model

Network: strongly r-connected di-graph



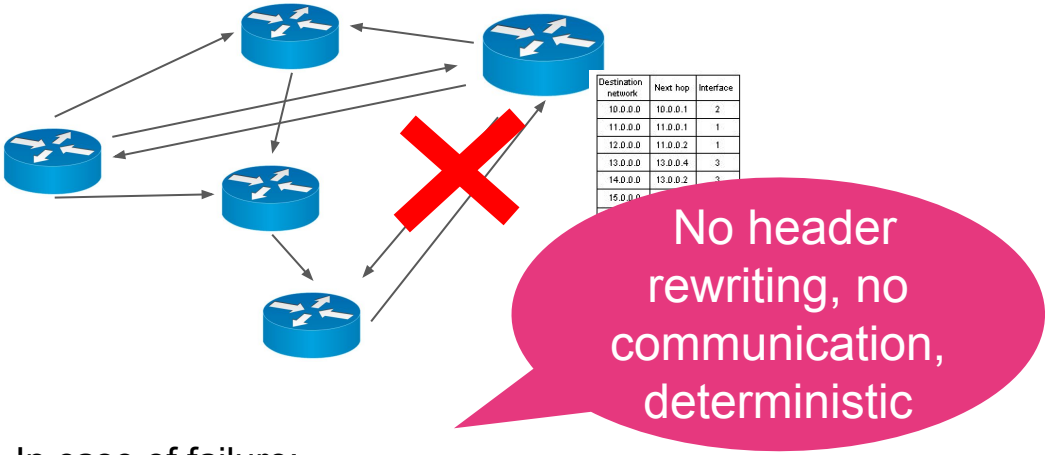
In case of failure:
static local re-routing based on

- SRC, DST, in-port
- incident failures

Model and Objectives

Model

Network: strongly r-connected di-graph



In case of failure:
static local re-routing based on

- SRC, DST, in-port
- incident failures

Objectives

Load
Maximum additional link utilization due to rerouting

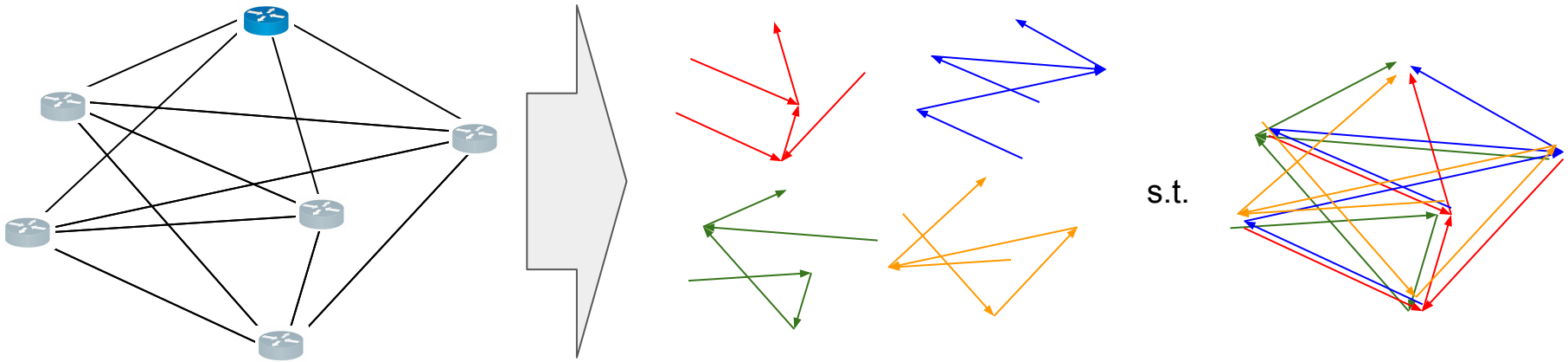
Stretch
Maximum additional hops due to rerouting

SRLG
Shared Risk Link Groups

Path independence
No shared intermediate nodes to destination

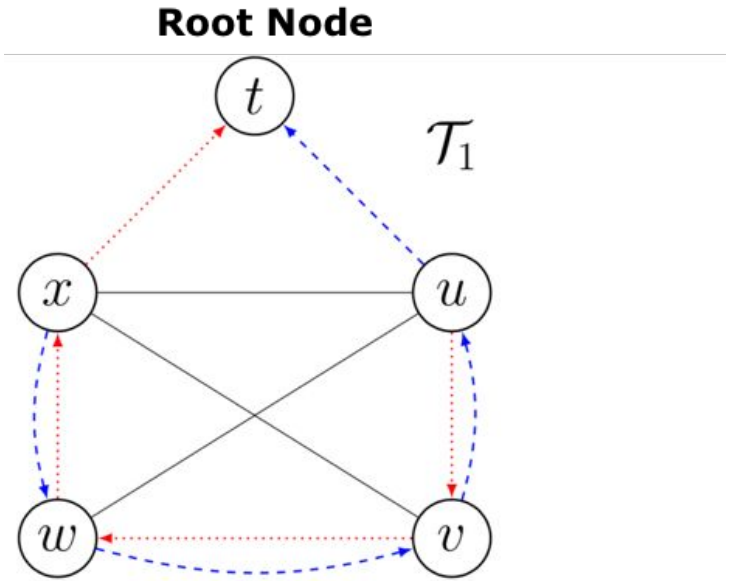
Arc-disjoint Arborescence Decomposition

- Arborescence = a rooted directed spanning tree
- Decomposition: union of r-arborescences uses each link at most once



Arborescence FRR

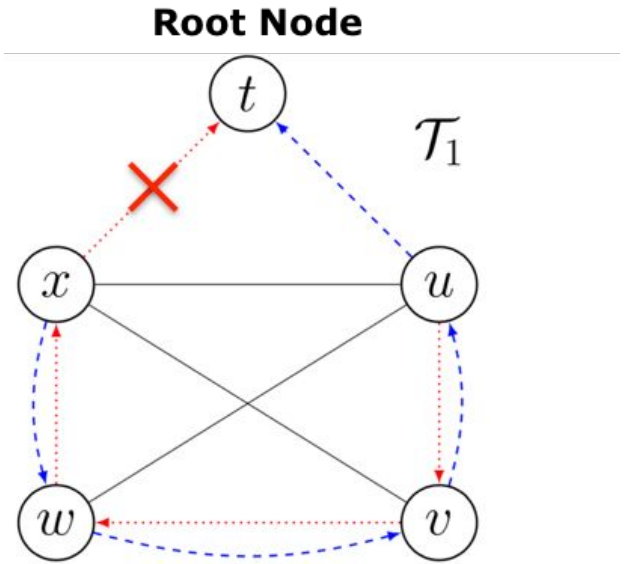
- Assign numbers to arborescences, pick arborescence 1
- Forward to next hop according to current arborescence

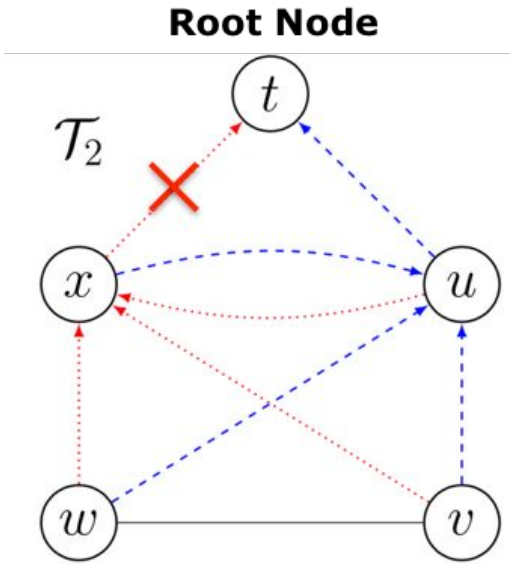


Arborescence FRR

Arborescence FRR

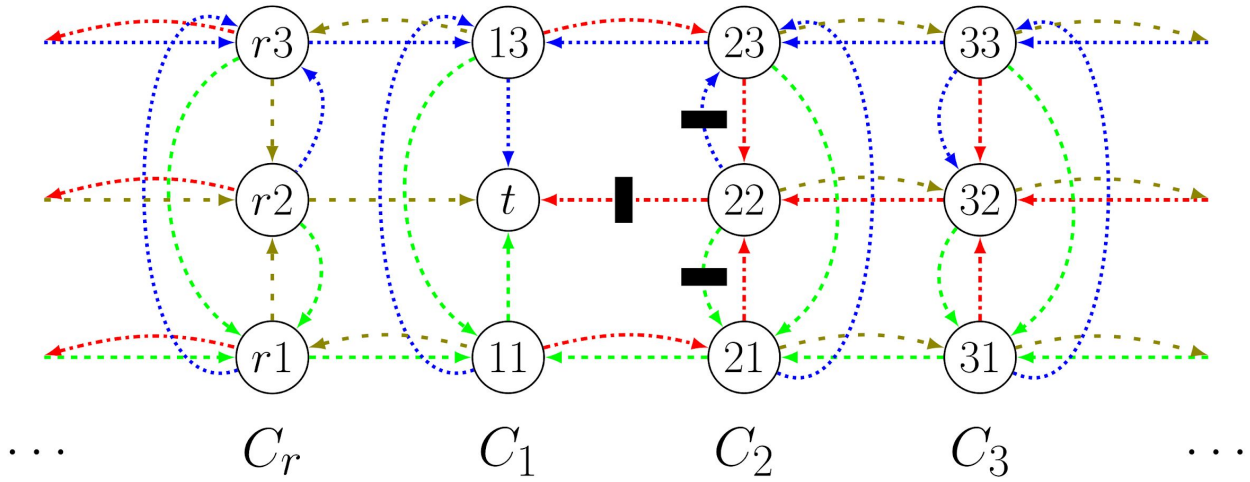
- Assign numbers to arborescences, pick arborescence 1
- Forward to next hop according to current arborescence
- If forwarding link is not available, use link of next arborescence





Decomposition influences length/load/..

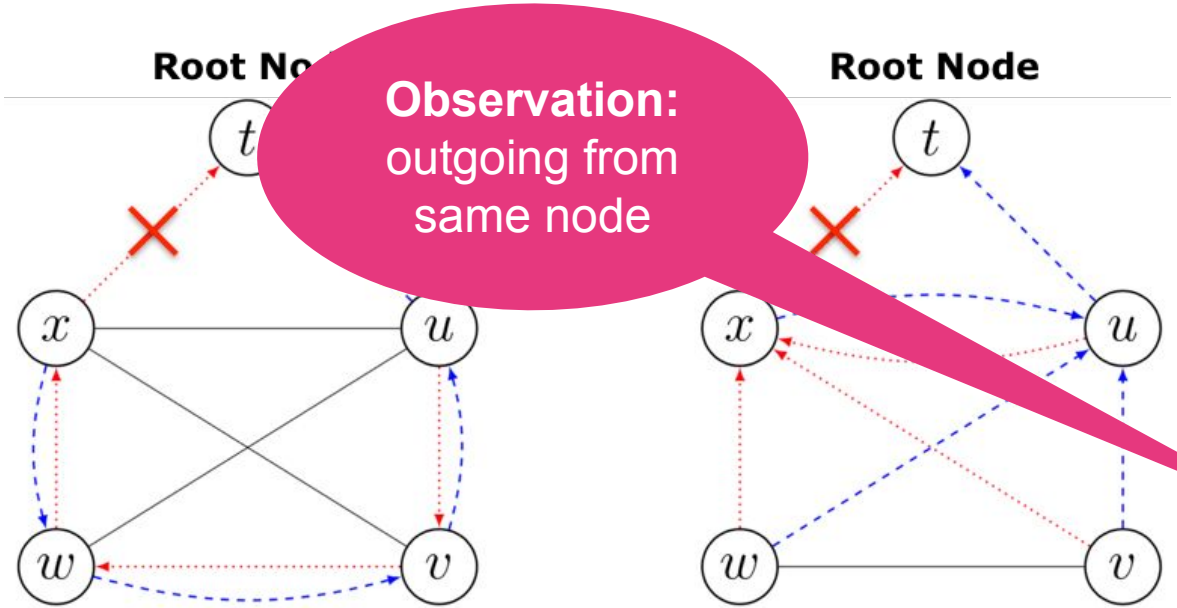
How good is Arborescence FRR?



Theorem 1: Deterministic local fast failover algorithms resilient to k − 1 failures, have competitive additive stretch of Ω(n/(k − 1))
(can be met by arborescence-based re-routing on donut graph).

Improve Decompositions

How to transform *T1* into *T2*?



Observation: outgoing from same node

(x,w) ↔ (x,u)

(u,v) ↔ (u,x)

…

Swapping Conditions

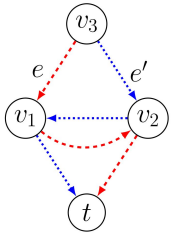
An arborescence swap *e=(u,v)* with *e'=(u,v')* is valid if

I. e ∈ Ti, e' ∈ Tj and
   - v′ is not on the path from v to the root in Tj
   - v is not on the path from v' to the root in Ti

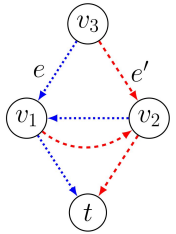
or

II. e ∈ Ti and e' not in any Tj and
   - v is not on the path from v' to the root in Ti

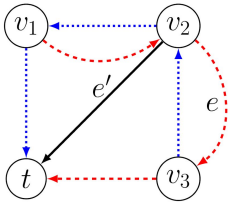
Observation: Validity computable in O(n)



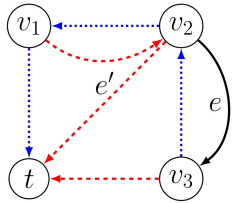
Before swapping



After swapping



Before swapping



After swapping

Post-Processing Algorithm

Decomposition

Objective function

Improved decomposition

Theorem 2.

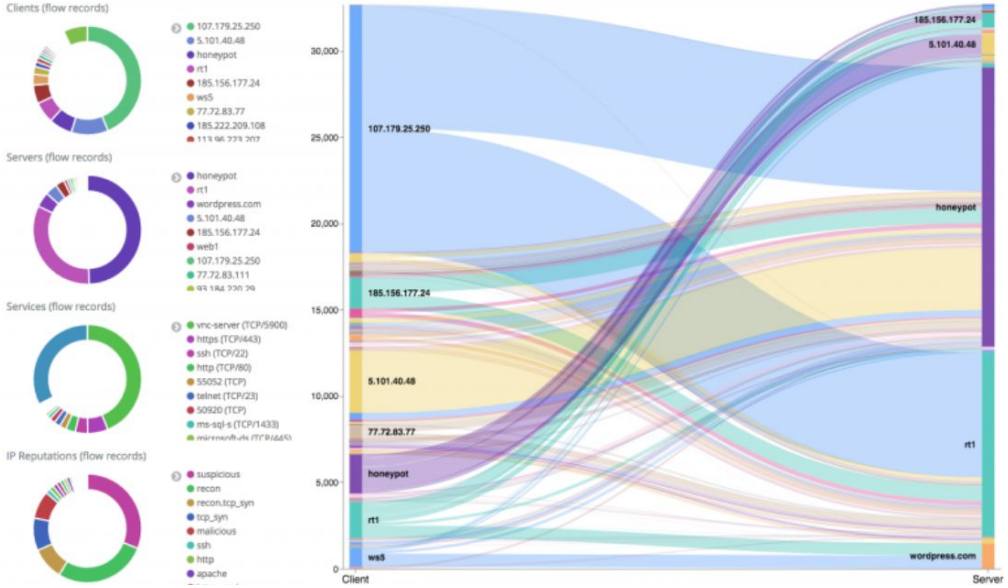
Post-processing algorithm never introduces cycles and always converges.

Case study 1

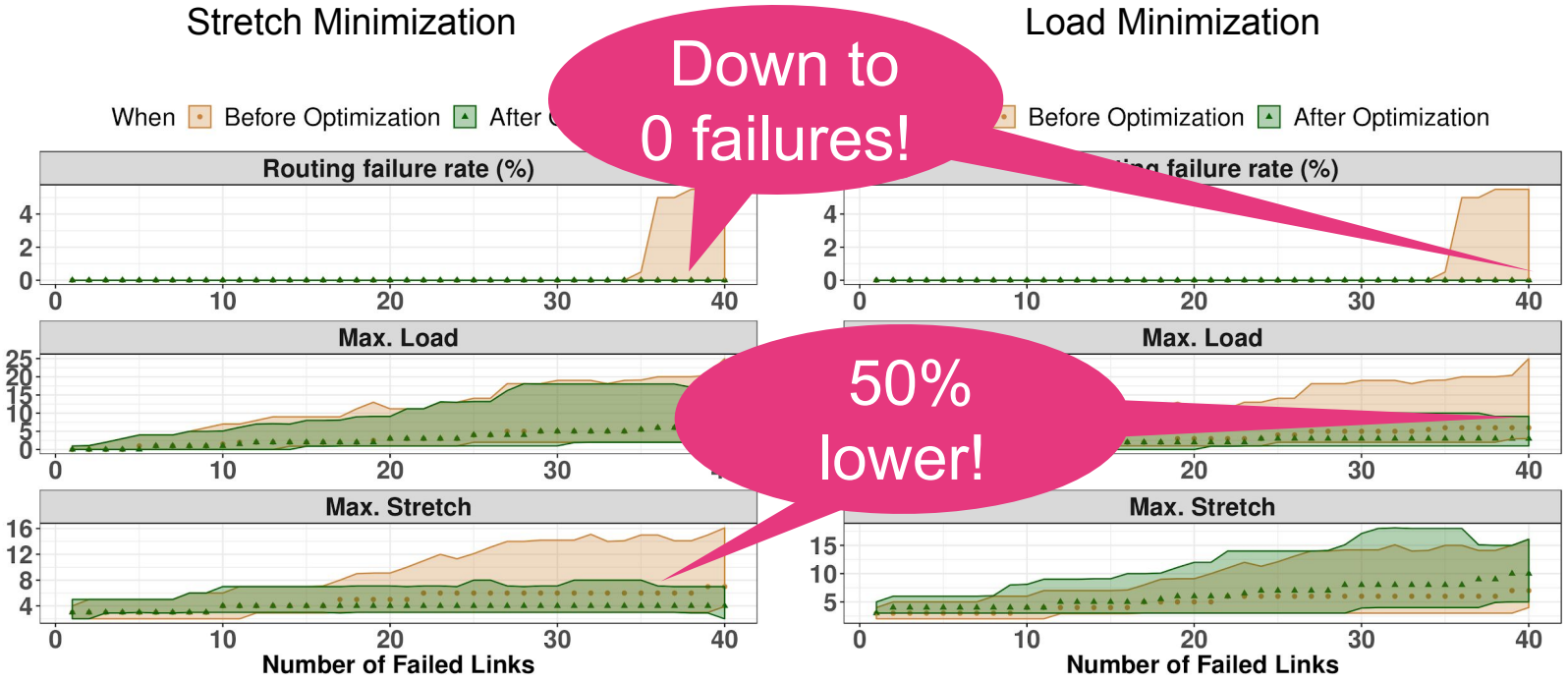
Traffic scenario optimization

- Flows differ in size and importance
- Links differ in failure probability

=> Minimize stretch/load of important flows given a failure model



Traffic Scenario



Case study 2

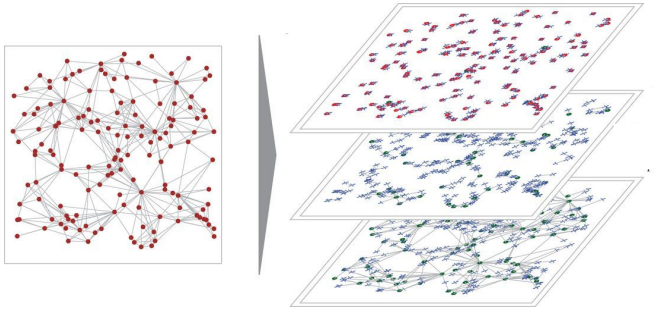
Direct decomposition optimization

- Shared Risk Link Groups (SRLG)

  => Links in SRLG in same arborescences

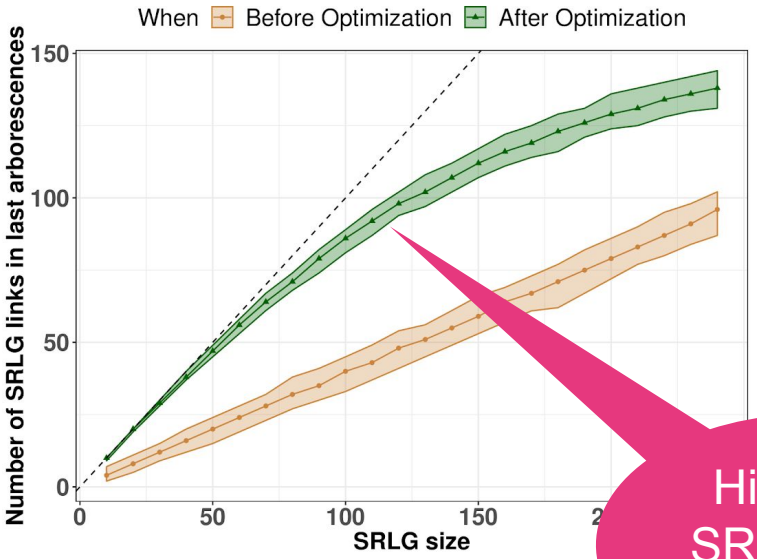
- Path independence

  => No shared intermediate nodes on routes to destination



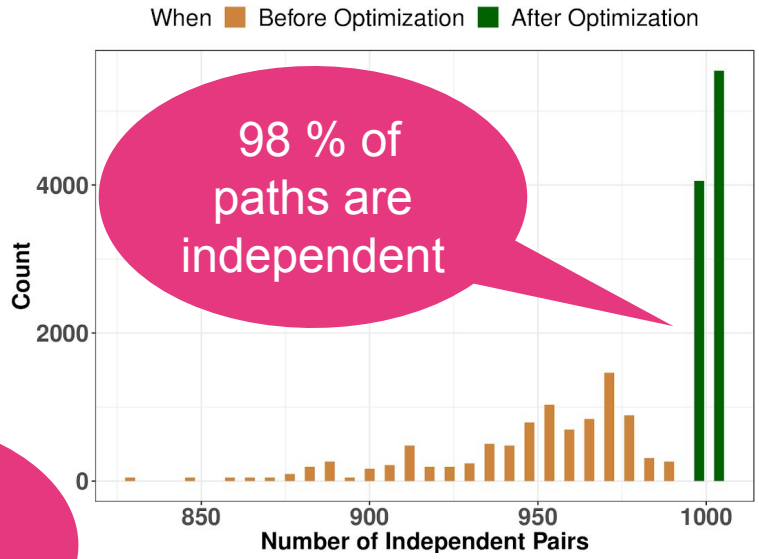
SLRG and Independence

SLRG



High % of SRLG links in last arbs

Independence



98 % of paths are independent

Conclusions

FRR to provide QoS in addition to basic connectivity

- FRR with arborescence decompositions can be asymptotically optimal wrt stretch
- Simple post-processing framework with convergence guarantee

  Case studies demonstrate applicability for stretch, load, independence, SRLG

Future work

- Bounds on improvement achieved
- Alternative post-processing strategies

Post-Processing Algorithm

```
Input: arborescence decomposition T, objective function
Output: improved decomposition

1.  improved := True
2.  while improved do
3.      improved := False
4.      for each node v do
5.          for all pairs of outgoing edges from v do
6.              if swapping condition met and objective function improves
7.                  swap edges in T
8.                  improved := True
```

Theorem 2.

Post-processing algorithm never introduces cycles and always converges.