Lancet: Better Network Resilience by Designing for Pruned Failure Sets

Yiyang Chang*†, Chuan Jiang*, Ashish Chandra*,
Sanjay Rao*, Mohit Tawarmalani*

*Purdue University, †Bytedance

This work was done when Yiyang Chang was at Purdue University

ACM Sigmetrics 2020

Challenges in Network Design

- Failures are important in designing wide-area networks
  - Inevitable [1, 2] and costly
- Network users desire high service level objectives (SLOs)
  - 99.99% or even 99.999%

[1] Gill, et al, Understanding network failures in data centers: Measurement, analysis, and implications. Sigomm 2011.

[2] Potharaju and Jain, When the network crumbles: An empirical study of cloud network failures and their impact on services, SOCC 2013.

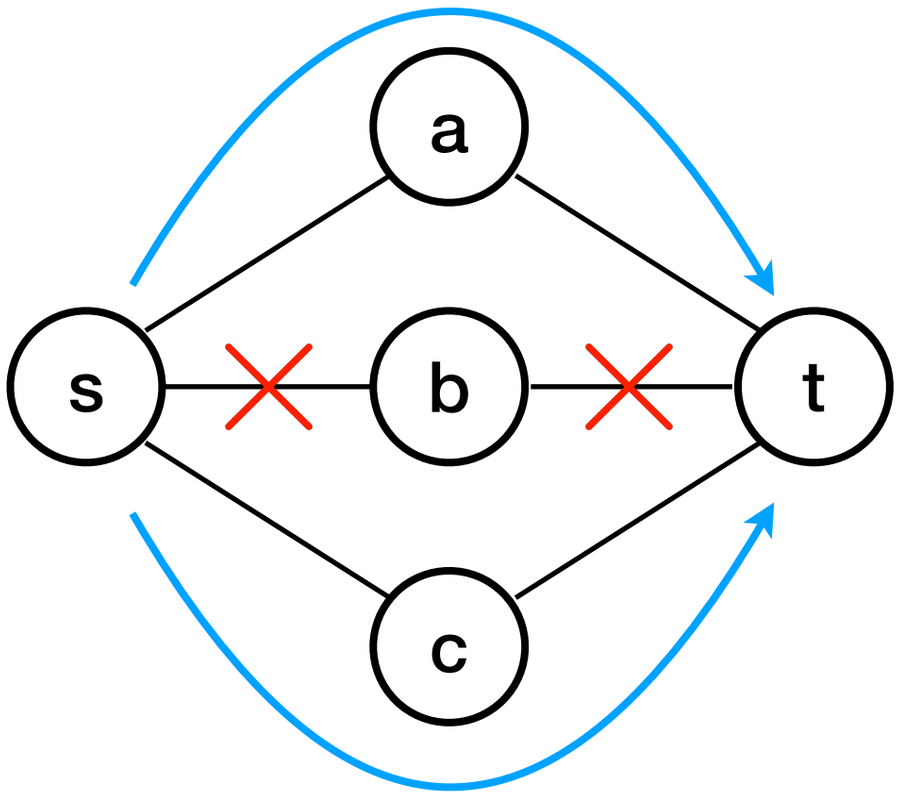
State-of-the-art in Network Design

- Key problem: how to design networks for such stringent requirements?

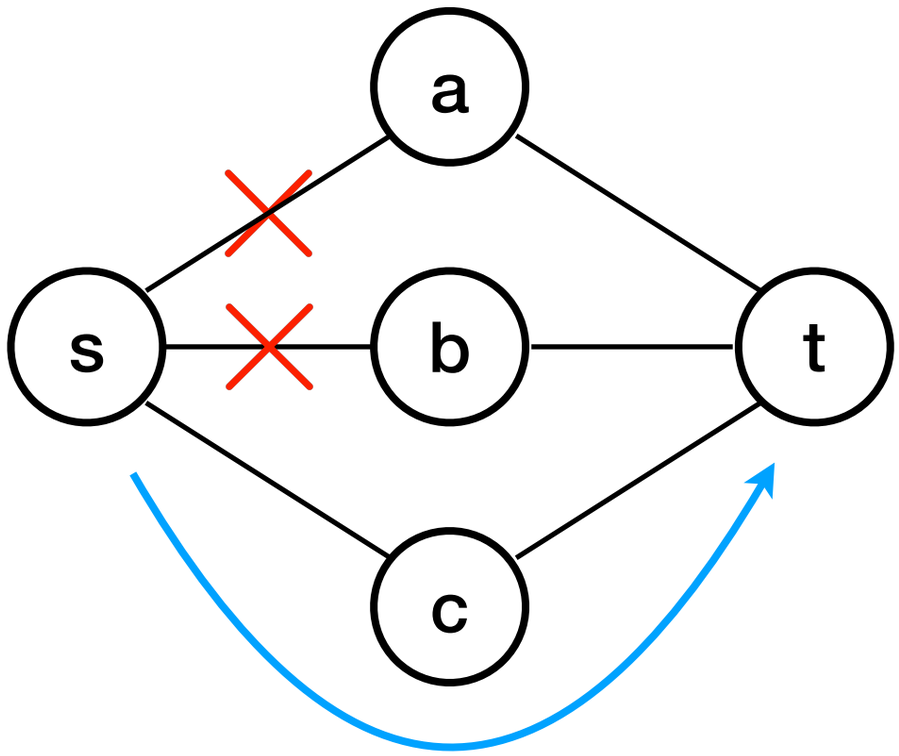
- State-of-the-art: Design for worst-case failure
  - Robust to all possible combinations of *f or fewer* failures
    - A weak point: If a single *f*-failure scenario cannot be tackled, forced to design for *f-1* failures only
- Examples: R3 (Wang, et al, Sigcomm 2010), FFC (Liu, et al, Sigcomm 2014)

Lancet - Beyond Worst-case

- Designing for worst-case may be conservative
- Can we design for most *f*-failure scenarios when designing for all is not possible?



Good 2-failure scenario



Bad 2-failure scenario

Link Capacity	1 unit
Demand from s to t	2 units

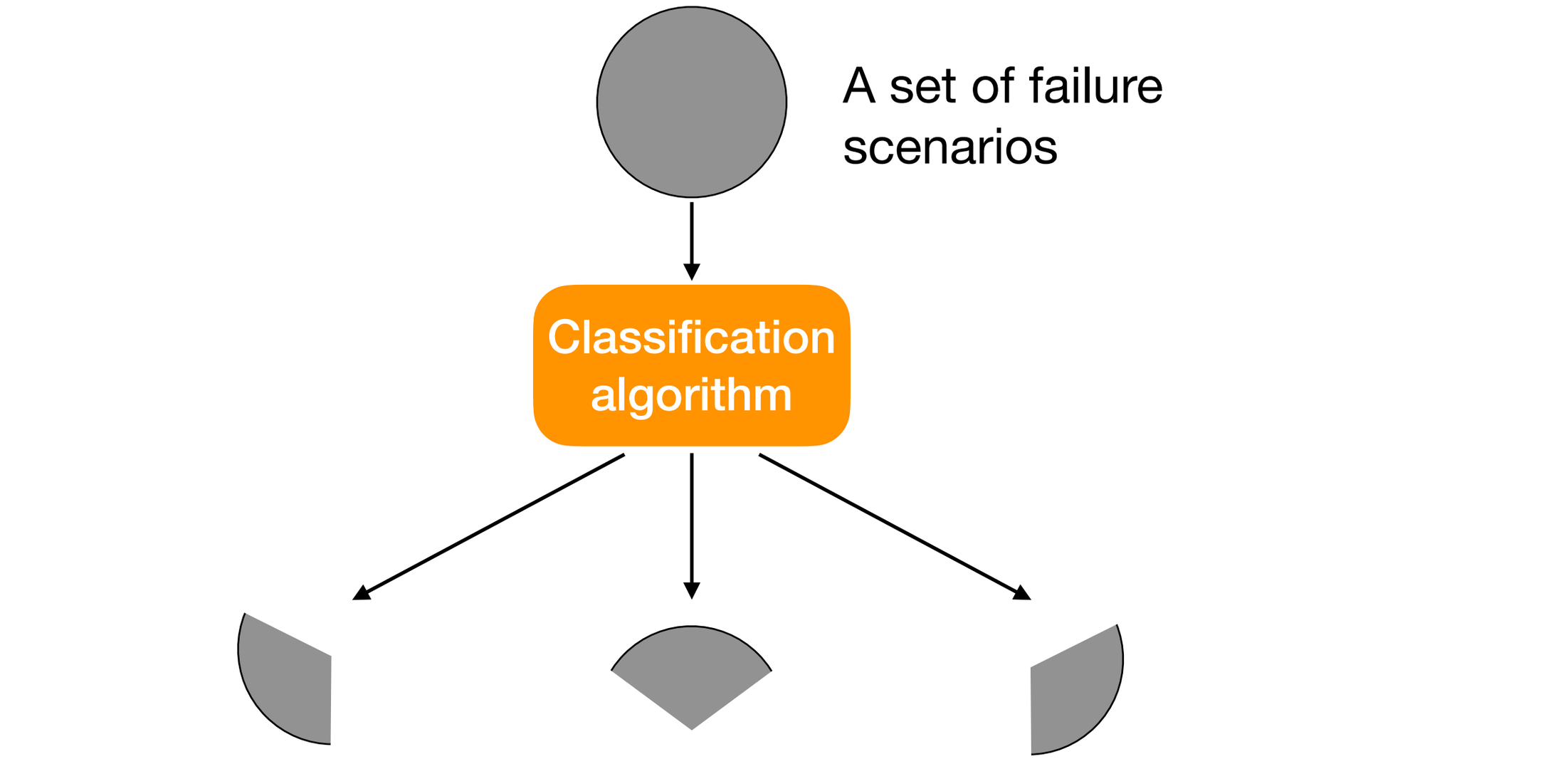
Lancet - Contributions

- New approach to designing protection routing
  - For most failure scenarios, when designing for all not possible
- Key components
  - Novel divide-and-conquer algorithm to efficiently identifies failure scenarios which a network can intrinsically handle
  - Provides a compact representation of these scenarios
  - Linear program (LP) approach to designing protection routing that exploits this compact representation
    - Cuts design time from > 18 hours to 10 seconds for a real-world topology
- Validations on real-world network topologies show Lancet's promise

Determine Which Scenarios to Design for

- How to determine which scenarios to design for?
  - Observation: Any routing scheme cannot perform better than an ideal scheme. An ideal scheme routes using multi-commodity flow
  - Exclude all bad scenarios with the ideal scheme
  - Design for the rest of the failure scenarios
- How to find which scenarios can be handled by the ideal scheme?
  - A divide-and-conquer algorithm to classify which failures can and cannot be handled

Lancet Classification Algorithm



Classification Algorithm in Operation

f = 0

Do all certify?

Yes.

Prune

Classification Algorithm in Operation

f = 0 f = 1

Do all certify?

Yes.

Prune

Classification Algorithm in Operation

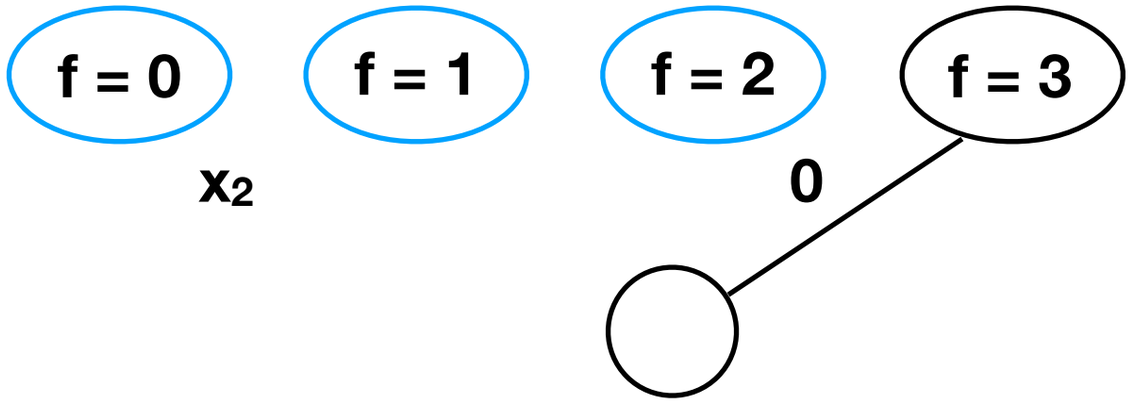
f = 0 f = 1 f = 2

Do all certify?

Yes.

Prune

Classification Algorithm in Operation



Do all certify?

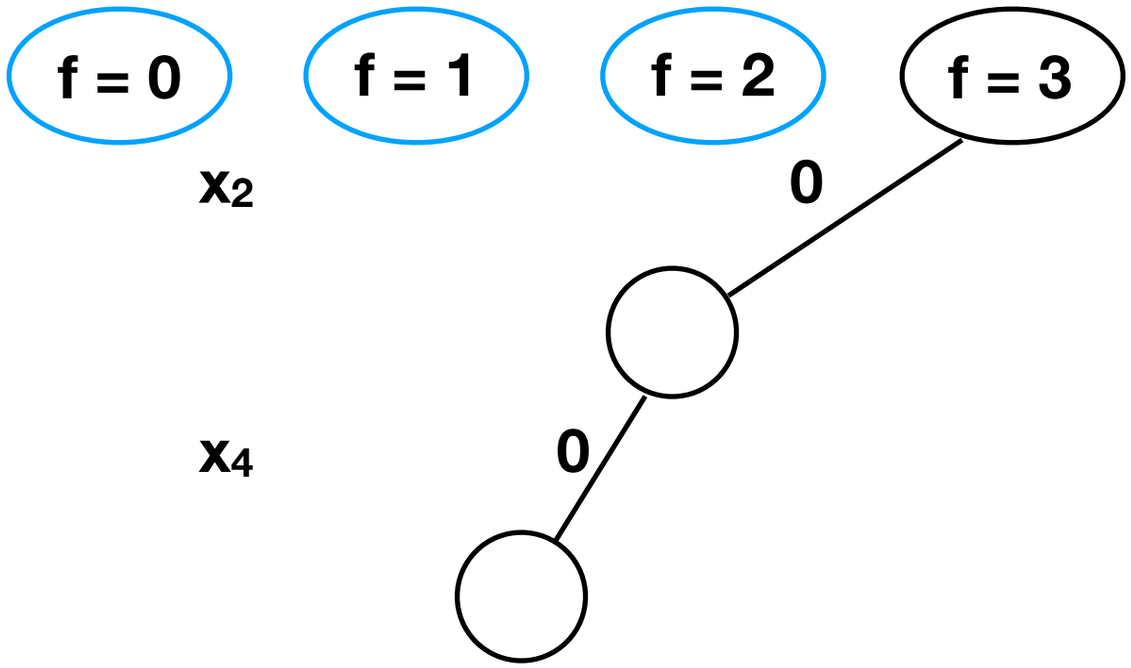
No.

Do all violate?

No.

Partition scenarios

Classification Algorithm in Operation

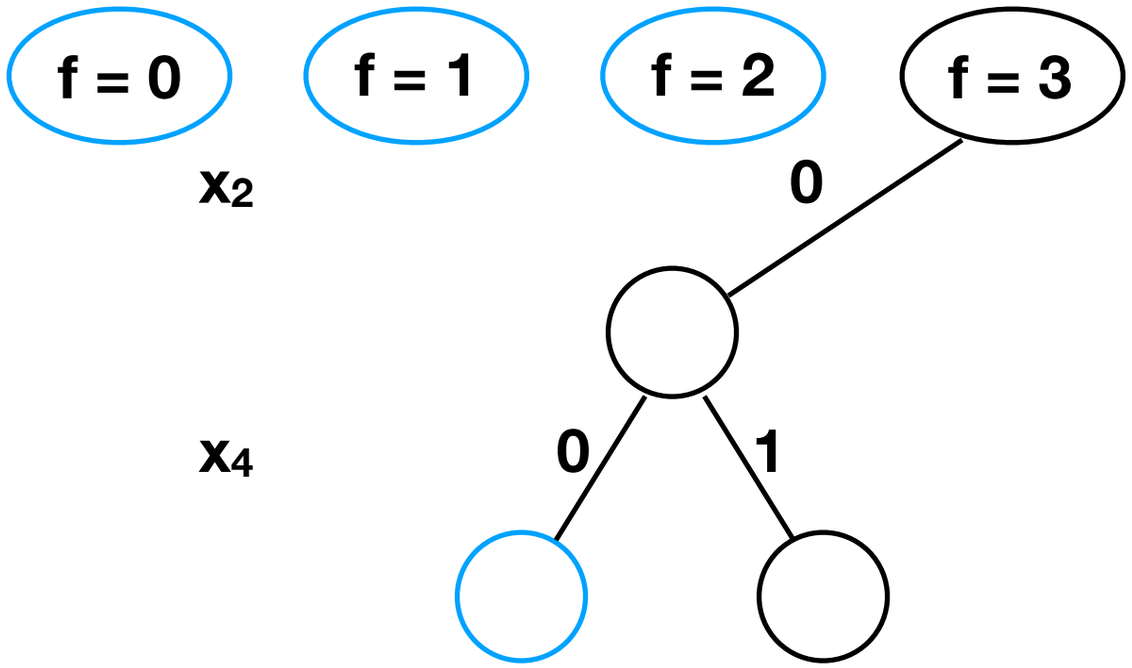


Do all certify?

Yes.

Prune

Classification Algorithm in Operation



Do all certify?

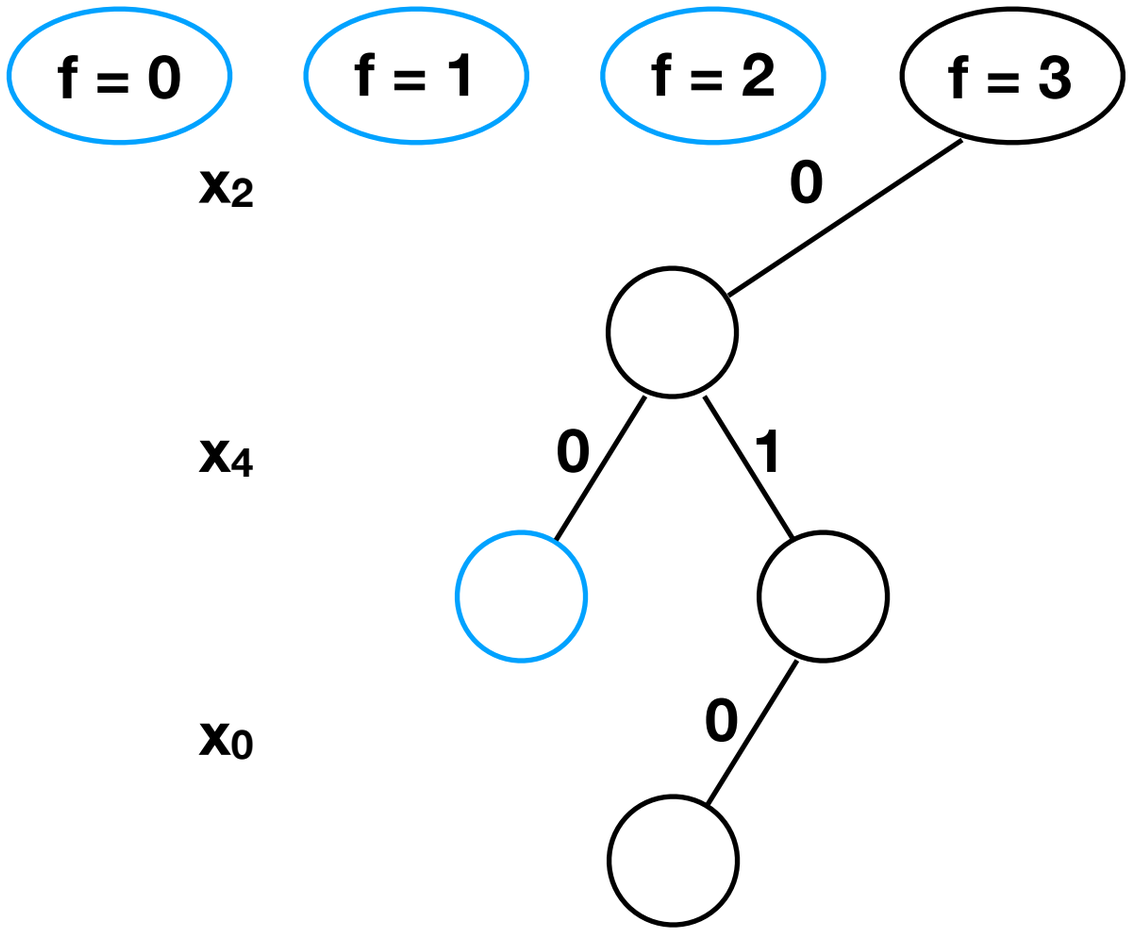
No.

Do all violate?

No.

Partition scenarios

Classification Algorithm in Operation

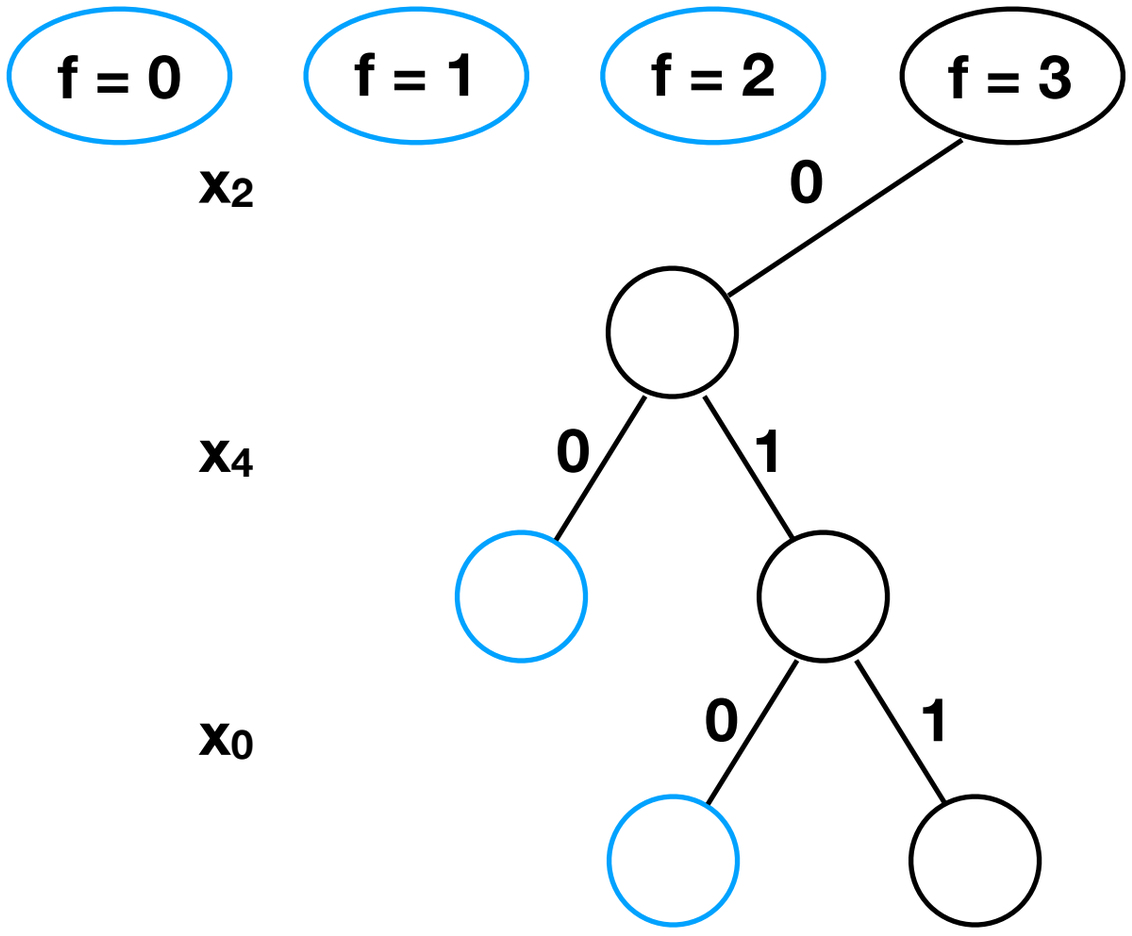


Do all certify?

Yes.

Prune

Classification Algorithm in Operation



Do all certify?

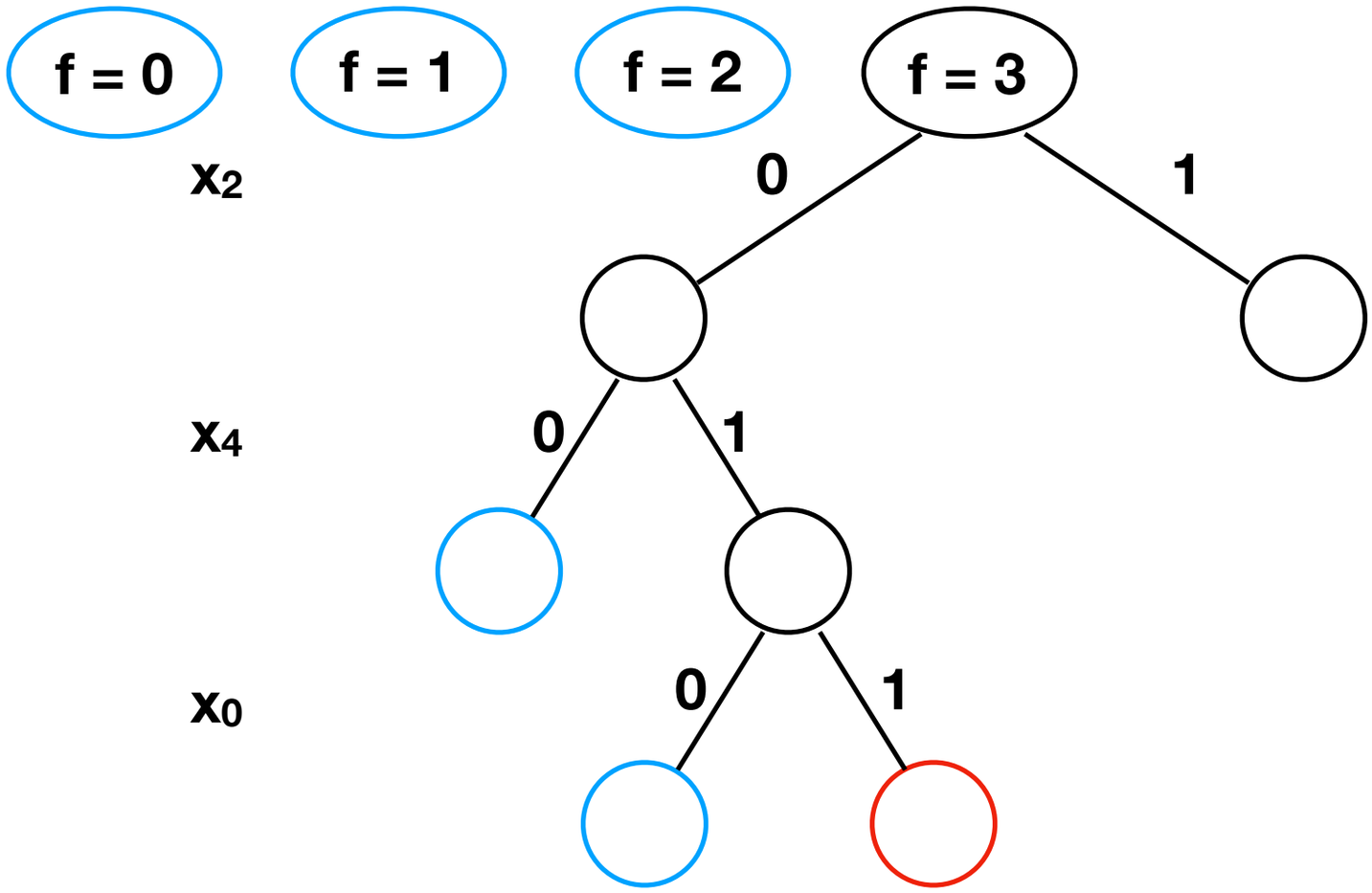
No.

Do all violate?

Yes.

Prune

Classification Algorithm in Operation



Do all certify?

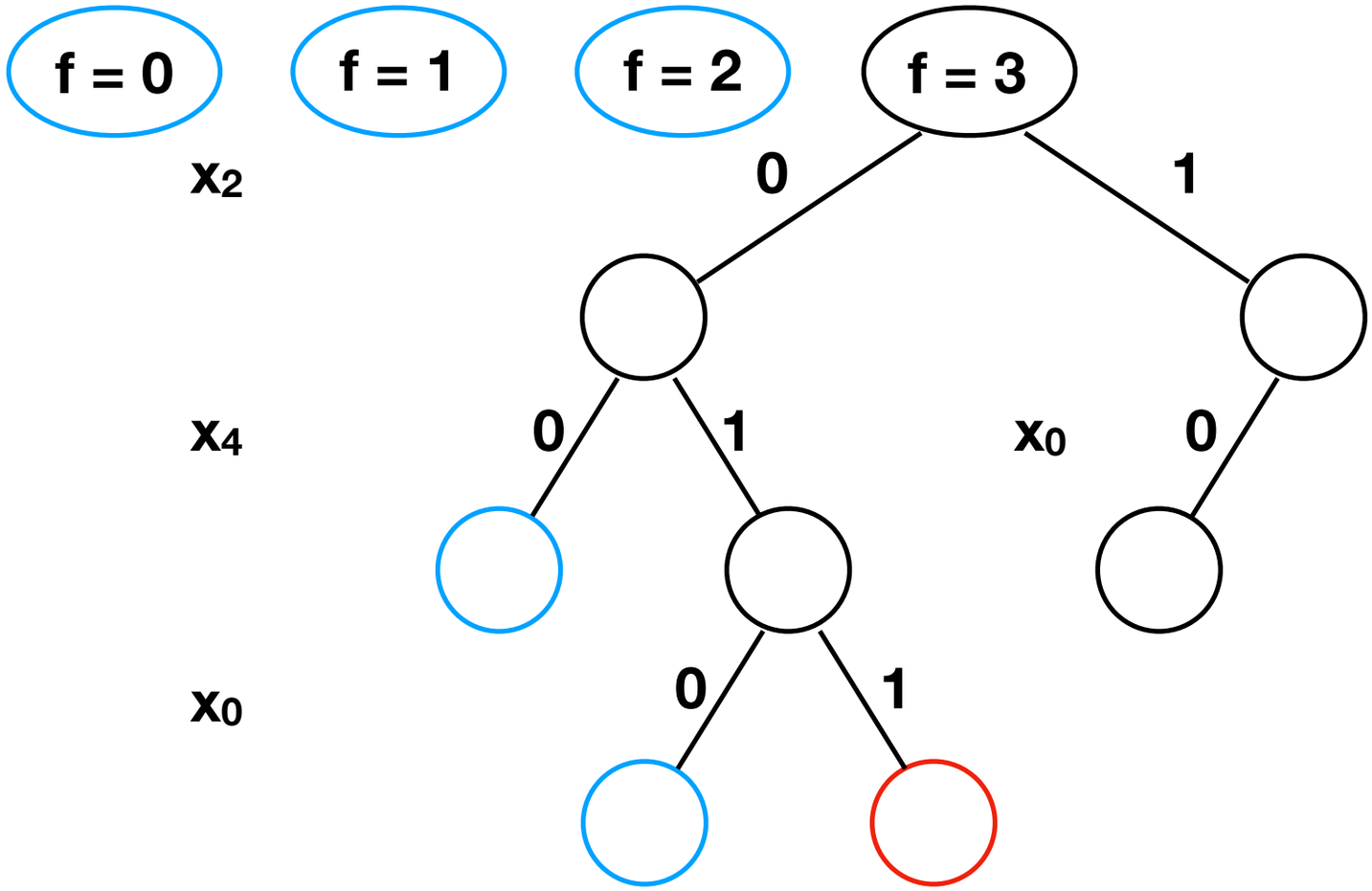
No.

Do all violate?

No.

Partition scenarios

Classification Algorithm in Operation

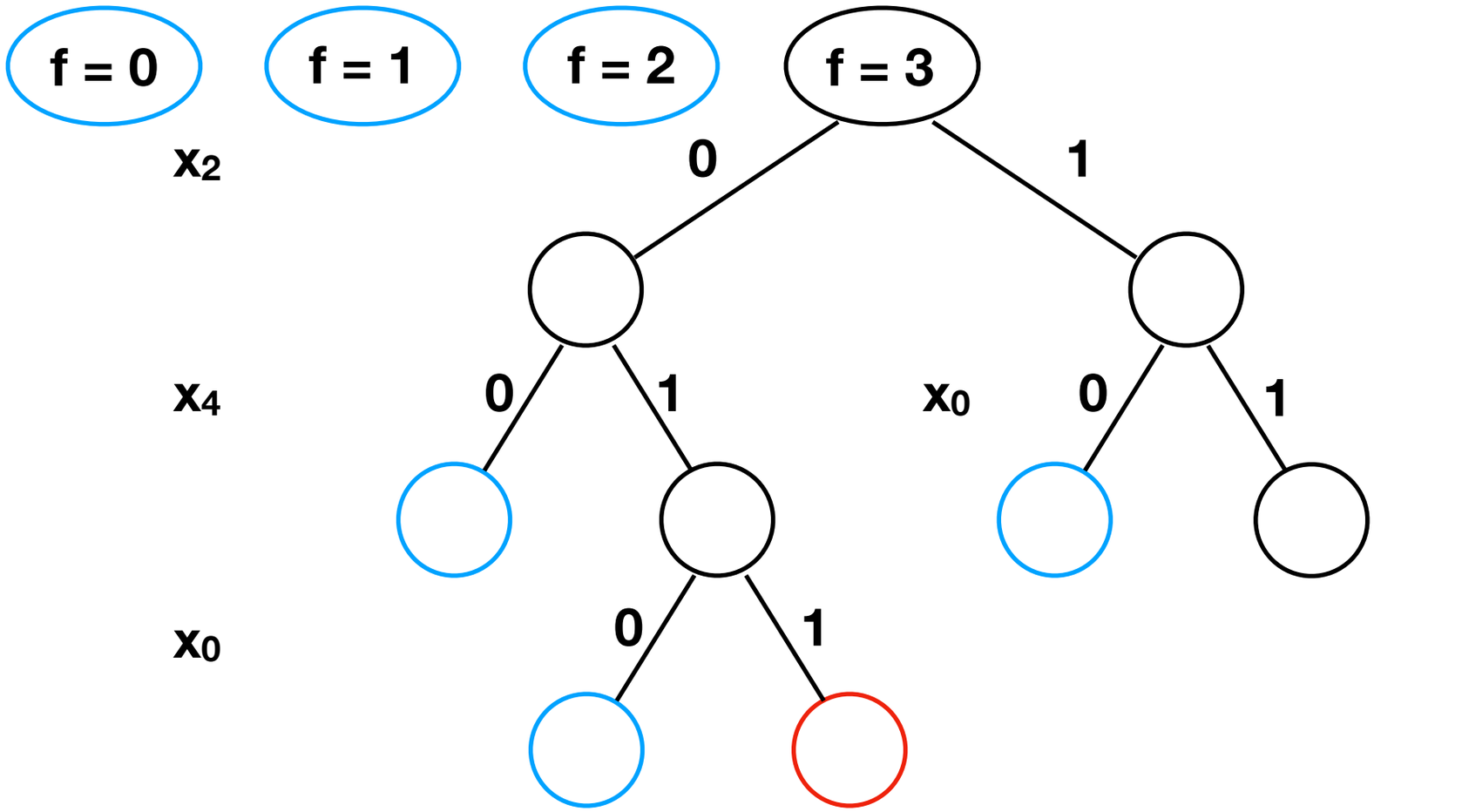


Do all certify?

Yes.

Prune

Classification Algorithm in Operation



Do all certify?

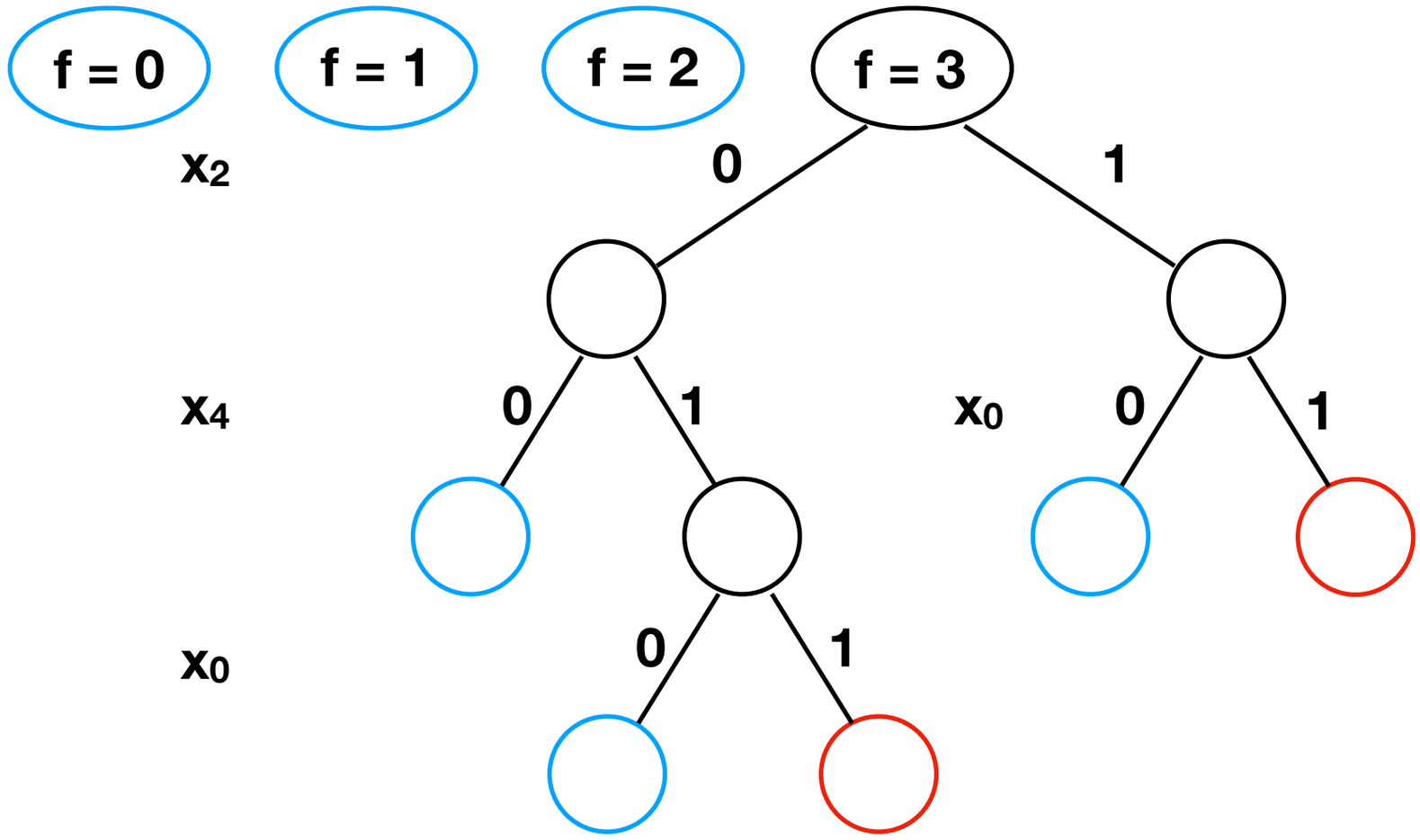
No.

Do all violate?

Yes.

Prune

Classification Algorithm in Operation



Done.

Key procedures

- *DoAllCertify()*
- *DoAllViolate()*
- Partitioning strategy

Keys for Tractable Classification

- *DoAllCertify(A)*
  - We show it is NP-complete
  - Instead, get a conservative bound
  - Doesn't affect correctness
- *DoAllViolate(A)*
  - Simple feasibility LP to test if there is a good failure scenario
- Partitioning strategy
  - Heuristic to choose a link *l* that fails in many bad scenarios

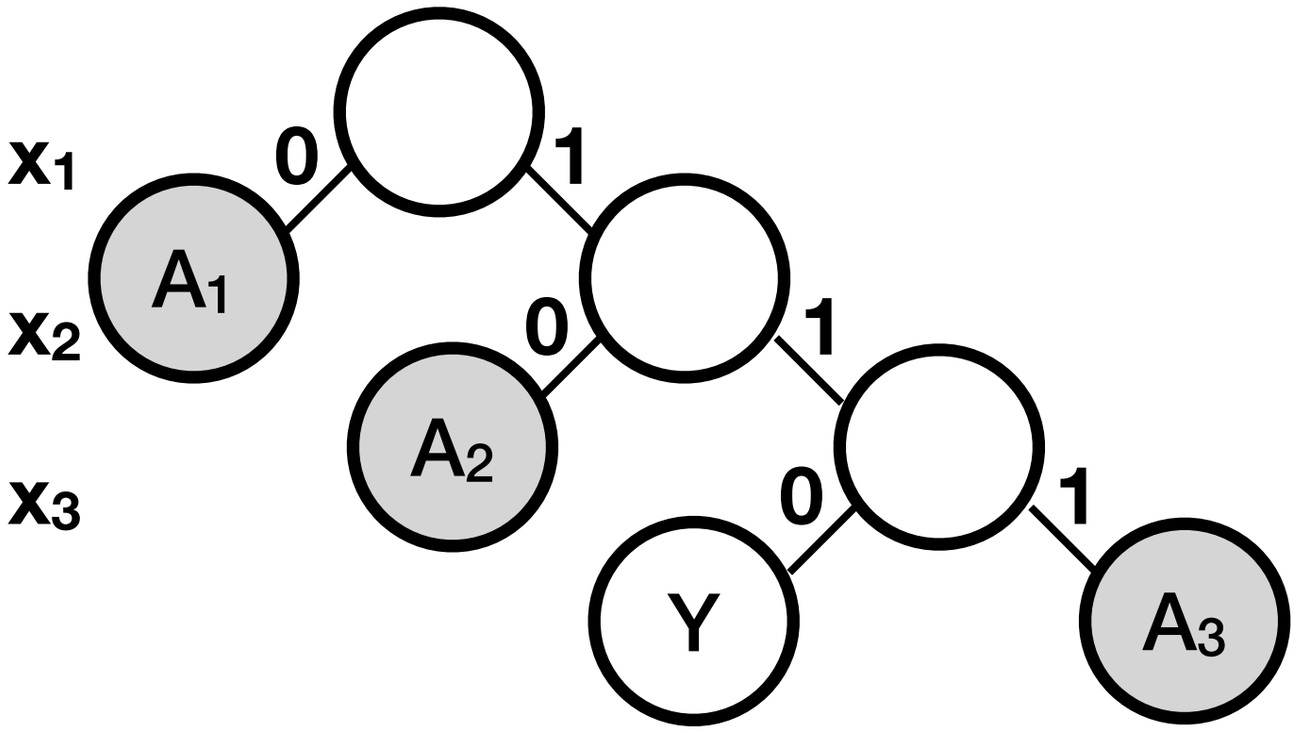
Algorithm 1: Lancet classification.

Function *Lancet()*
- SSets $\longleftarrow [F_f, F_{f-1}, \ldots, F_1, F_0]$
- pass_set, fail_set $\longleftarrow \emptyset, \emptyset$
- **while** SSets **do**
  - $A \longleftarrow$ SSets.*pop*()
  - *Classify*(*A*, pass_set, fail_set, SSets)

Function *Classify*(*A*, pass_set, fail_set, SSets)
- **if** *DoAllCerify(A)* **then**
  - pass_set.*add*(*A*)
- **else if** *DoAllViolate(A)* **then**
  - fail_set.*add*(*A*)
- **else**
  - $A_1, A_2, \ldots, A_n \longleftarrow$ *Partition*(*A*)
  - SSets.*extend*($[A_1, A_2, \ldots, A_n]$)

Compact Representation of Failure Sets

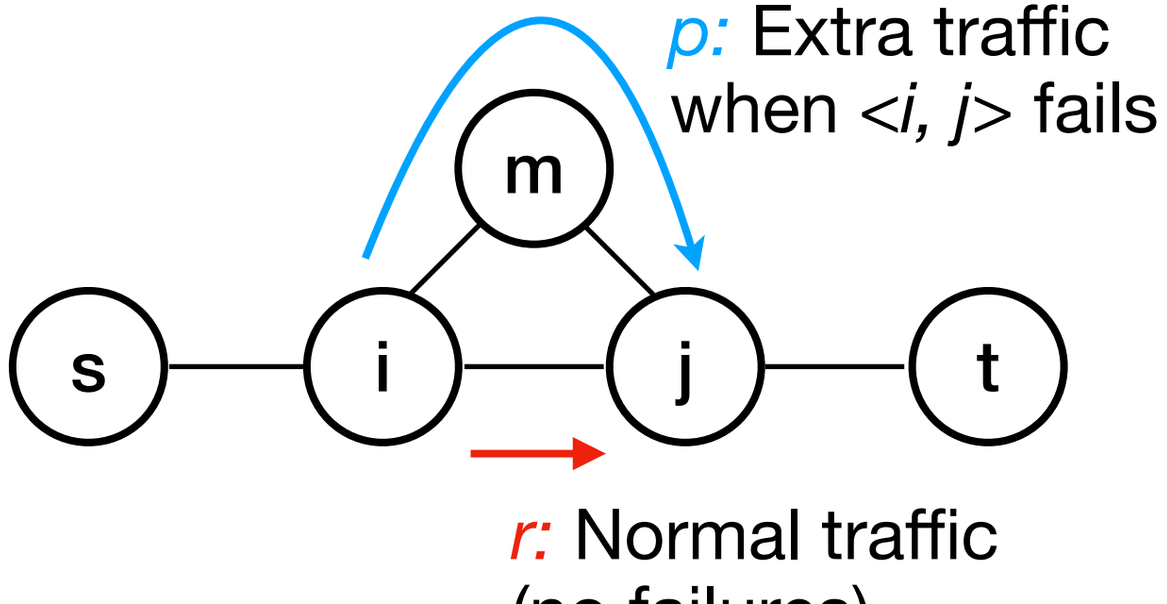
- Two ways to represent failure scenarios
  - A_1, A_2, and A_3 as 3 sets
  - 161k+ separate failure scenarios
- The classification algorithm naturally generates the first representation
- Next we will see why the first representation is better



- Sets of 3-failure scenarios of a 100-link network
- A_1, A_2, and A_3 certify
- *Y* is undecided

Protection Routing Design

- Link-based protection routing
  - Provisions bypass paths to protect against each failure scenarios
  - Achieved using (H), generalizing a state-of-the-art scheme [1]
- Issues with existing protection routing schemes
  - Only work if *X* is all f failures
  - If worst-case $U > 1$, we are forced to design for $f - 1$ failures
- What we want: Design for most *f* failures if not all



$$\text{(H)} \quad \min_{r,p,a,U} \quad U$$

$$\text{s.t. } r_{st} \text{ is a unit flow from } s \text{ to } t. \quad \forall s,t \in V$$

$$p_l \text{ is a flow of } a_l \text{ from } i \text{ to } j. \quad \forall l \in E, l = \langle i,j \rangle$$

$$\forall x \in X, e \in E,$$

$$\sum_{s,t} d_{st} r_{st}(e) + \sum_{l \in E} x_l p_l(e) \le U c_e (1 - x_e) + a_e x_e$$

$$a_e \ge 0 \quad \forall e \in E; \quad U \ge 0$$

[1] Wang, et al, R3: Resilient routing reconfiguration, Sigcomm 2010

Protection Routing Design with Excluded Scenarios

- Two ways to implement the capacity constraints (circled in red)
  1. Enumerate constraints, one for each failure scenario x
  2. Impose the constraint for a union of failure sets, each one represented using LP duality
- Our approach (2nd above) is more compact
  - Since number of sets can be exponentially smaller than the number of failure scenarios

(H) $\min_{r,p,a,U} \quad U$

s.t. r_{st} is a unit flow from s to t. $\quad \forall s,t \in V$

p_l is a flow of a_l from i to j. $\quad \forall l \in E, l = \langle i,j \rangle$

$\forall x \in X, e \in E,$

$$\sum_{s,t} d_{st} r_{st}(e) + \sum_{l \in E} x_l p_l(e) \le U c_e (1 - x_e) + a_e x_e$$

$a_e \ge 0 \quad \forall e \in E; \quad U \ge 0$

Summarizing Design with Lancet

- Step 1: Reformulate (H) to an LP to handle arbitrary sets of failure scenarios
- Step 2: Determine which failure scenarios (represented in failure sets) to include with the classification algorithm
- Step 3: Leveraging the LP in Step 1, design a protection routing scheme for failure sets discovered in Step 2

Evaluations

- Real topologies

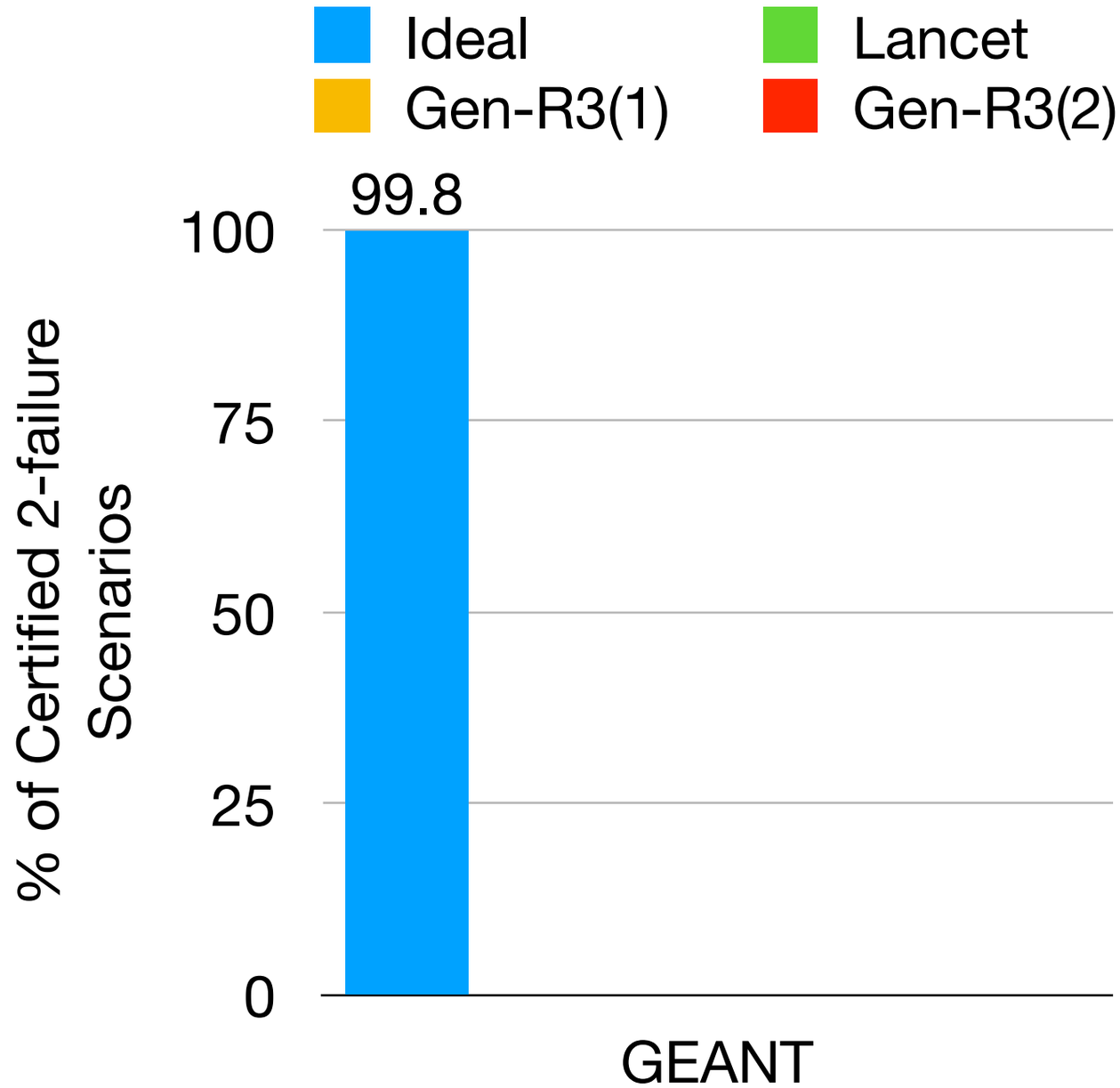
Network	# of Nodes	# of Edges	# of sub-links
Abilene	11	14	2
GEANT	32	50	2
Deltacom	103	151	2
ION	114	135	2

- Partial failure model
  - All links comprises 2 sub-links
- Synthetic traffic matrix: Gravity model [1]
- Environment: single-threaded on a 3.00GHz Intel Xeon CPU
- Implemented in Python and Gurobi 8.0

[1] Yin Zhang, et al. Network anomography. IMC 2005.

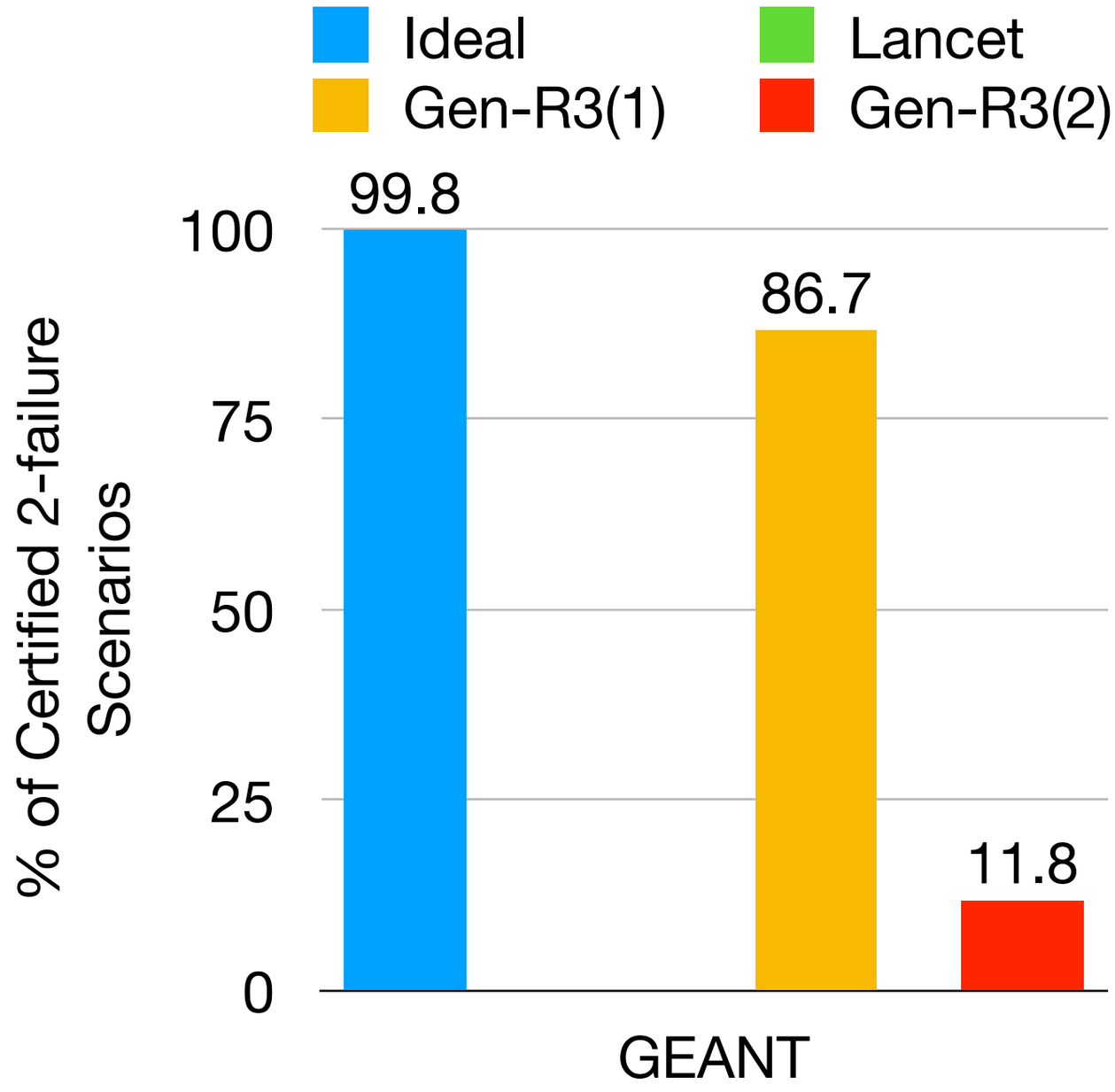
Design with Lancet

- The ideal scheme handles 99.8% of the 2-failure scenarios for GEANT



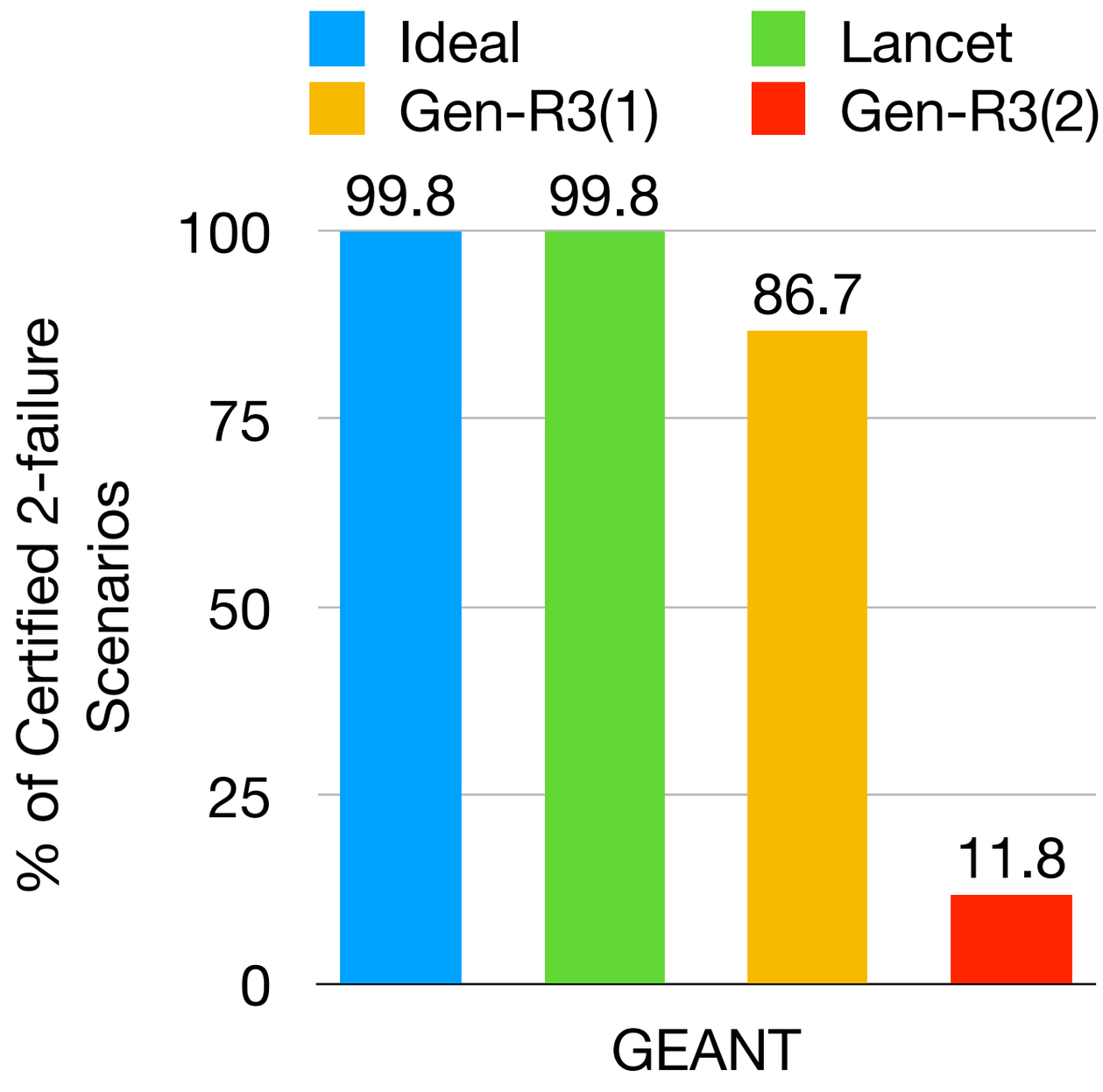
Design with Lancet

- Gen-R3 (f): the protection routing design obtained by optimizing worst-case *f*-failure scenarios
- Takeaway: Large performance gaps exist between Gen-R3 schemes and the ideal scheme

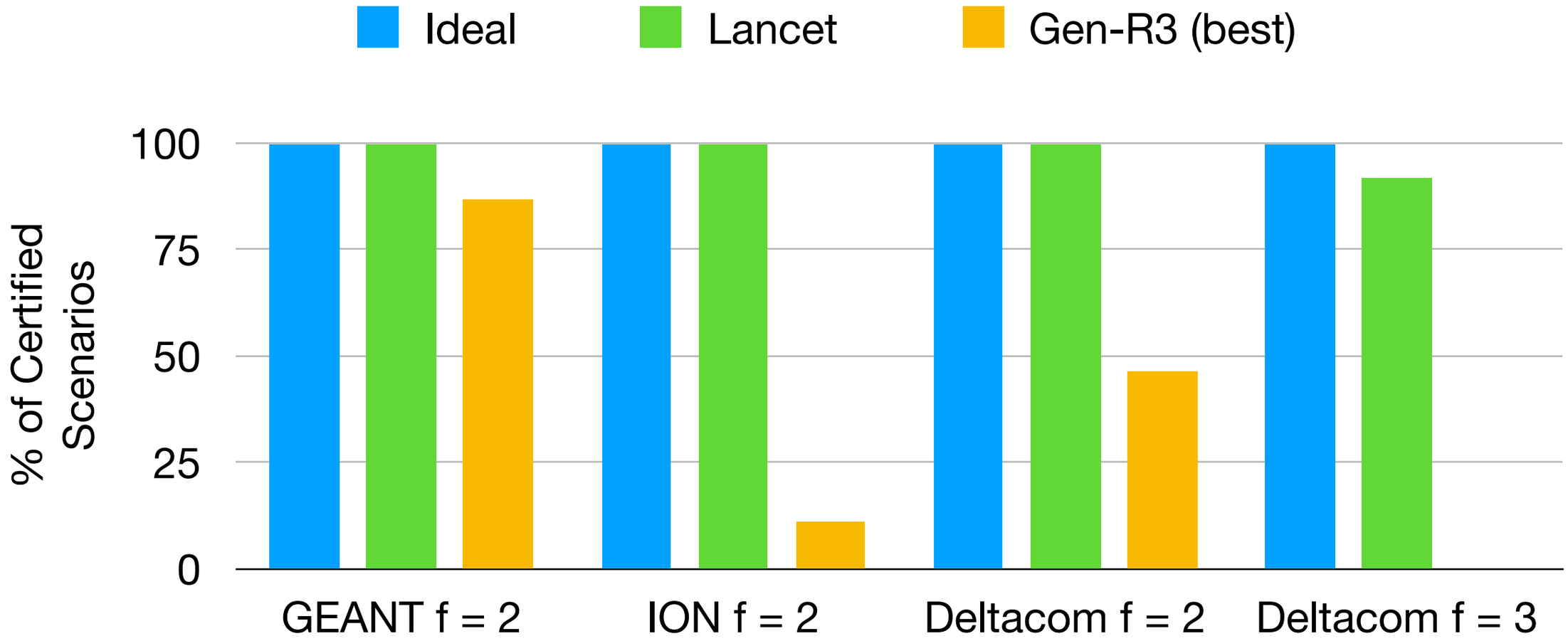


Design with Lancet

- Lancet: protection routing designed with Lancet by excluding bad failure scenarios
- Takeaway: Lancet bridges the performance gap, reaching optimal for GEANT



Design with Lancet on Larger Networks

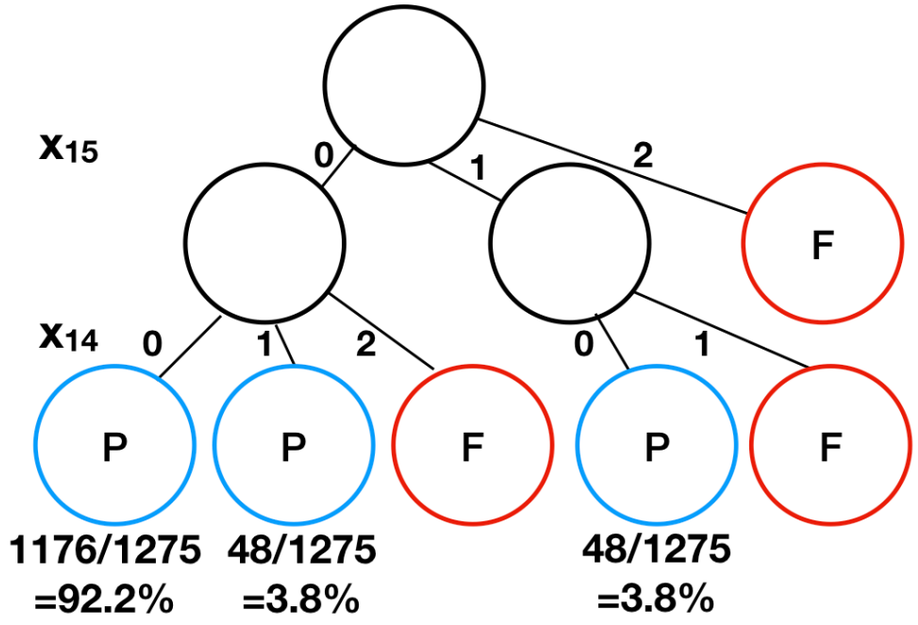


- Gen-R3 (best): the Gen-R3 (f) that gives the best result
- Takeaway: Lancet achieves much better performance than Gen-R3 (best) and is close to optimal

Compactness of Failure Set Representation

- Lancet represents a large number of failure scenarios in a small number of failure sets
- Enables tractable designs of protection routing

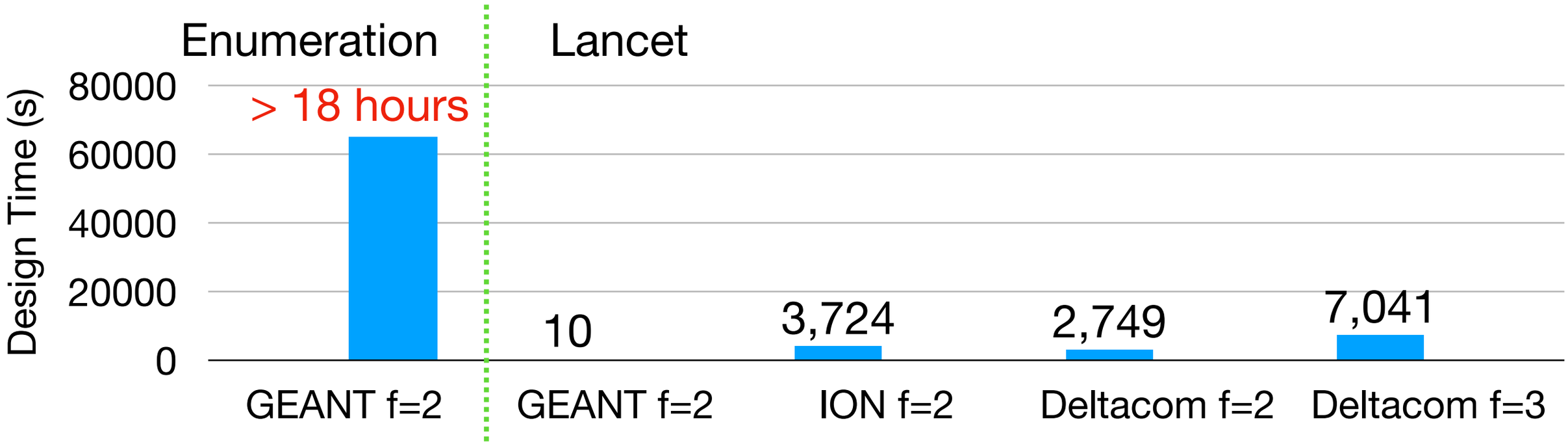
Topology (# of failures)	# of sets	# of scenarios
GEANT (2)	3	1272
ION (2)	5	9172
Deltacom (2)	6	11,465
Deltacom (3)	3	466,486



GEANT

Design Time with Lancet and Enumeration

- For a moderate-sized network GEANT
  - Lancet reduces design time from > 18 hours to 10 seconds
- Makes it possible to handle large topologies in less than 2 hours



Extensions and Other Results

- Generalizations and extensions
  - Richer failure models
    - E.g., Shared-risk link group (SRLG)
    - Design to meet probability requirements
    - Multiple traffic demands
- Other results
  - Design with multiple traffic classes
  - Validations on SDN testbed

Conclusions

- Network design for worst-case failure is conservative
- Lancet, an algorithm that efficiently identifies failure scenarios the network can handle
- Lancet yields a compact representation of good failure sets
- Design using the compact representation performs close to ideal, while reducing the design time form > 18 hours to 10 seconds
- Evaluations and validations on real-world topologies show the promise of Lancet

Thanks!

Email your questions to

Yiyang Chang: yiyangchang1024@gmail.com
Sanjay Rao: sanjay@ecn.purdue.edu

Backup slides

Protection routing design with excluded scenarios

- Why do we want to design with excluded scenarios?
  - Hard to find a good design with existing approaches, if the worst-case failure scenario is infeasible to tolerate
  - Incentive to design for most rather than all *f*-failure scenarios

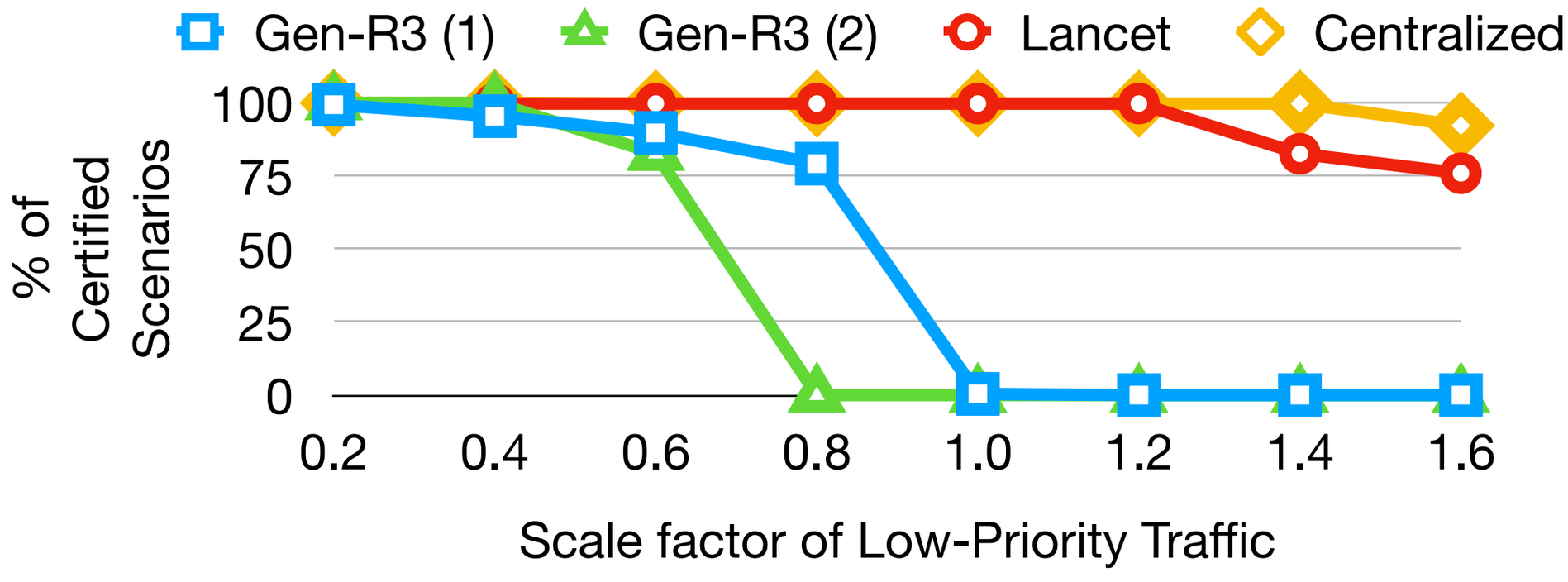
- Directly using formulation (H) is not scalable
  - O(*NE*) constraints, where *E* is the number of links, and *N* is the number of failure scenarios, which are often large (e.g., $N = \binom{E}{f}$ for all *f*-failure scenarios)
  - O(*NE*) dense constraints (i.e., constraints with large number of variables), largely impacting on computation time

Protection routing design with excluded scenarios

- The key is to reformulate (H) with compactly represented scenarios in the form of a union of *M* sets, where *M* is small compared to the number of failure scenarios
  - Reformulated (H) now has O(ME^2) constraints (*O(ME)* if each set has exactly one failure scenario)
  - Only *O(ME)* dense constraints
- Applies to partial link failure model
- Refer to the paper for the proofs and details on how the compact representation *exactly* reformulate (H)

Design with Multiple Traffic Classes

- Lancet applies to multiple traffic classes
  - Meet all high-priority, and as much low-priority traffic as possible.
  - Scale factor: after satisfying high-priority traffic, how much low-priority traffic is handled
- Split the original GEANT traffic matrix randomly into high- and low-priority
- **Takeaway**: Lancet performs nearly as well as Centralized (ideal). While it degrades moderately for the most stringent performance thresholds 1.4 and 1.6

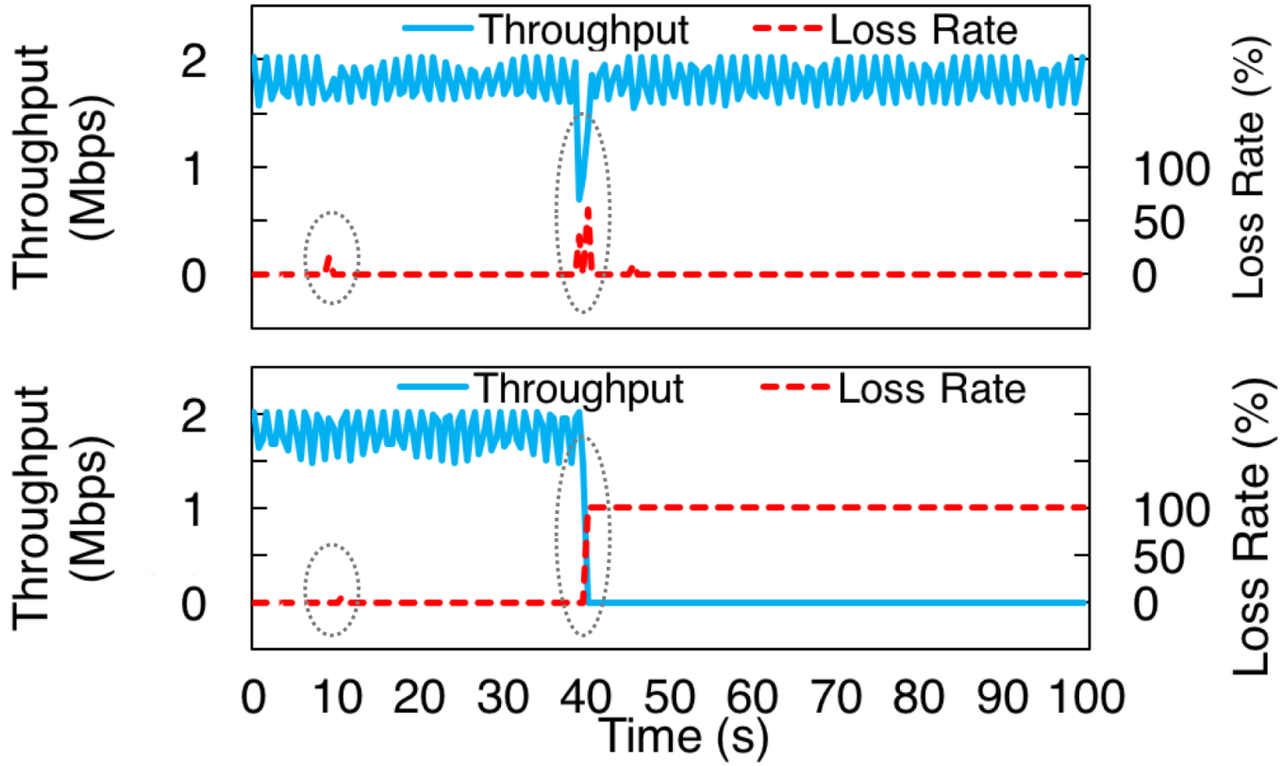


Validation on Testbed

- Emulation setup
  - Mininet 2.2 + OpenVSwitch 2.10 + OpenFlow 1.5
  - Abilene network, k = 1
    - MLU < 1 for single failures; MLU > 1 for two failures
- Protection routing implementation
  - Initial flow rules installed by a central controller
  - Failure information propagated by MPLS-label switching
  - Central controller updates protection routing on detecting failures

Validation on Testbed

- Experiment setup
  - Gen-R3: designed for *f = 1*
  - Lancet: designed for all *f <= 2* scenarios, excluding bad ones
  - 30 UDP flows with the same demands between source and destination
- **Takeaway**
  - Lancet tolerates the second link failure, but Gen-R3 fails to react
  - Reasoning: Gen-R3 resulted in two failed links mutually using each other to protect against their respective failures

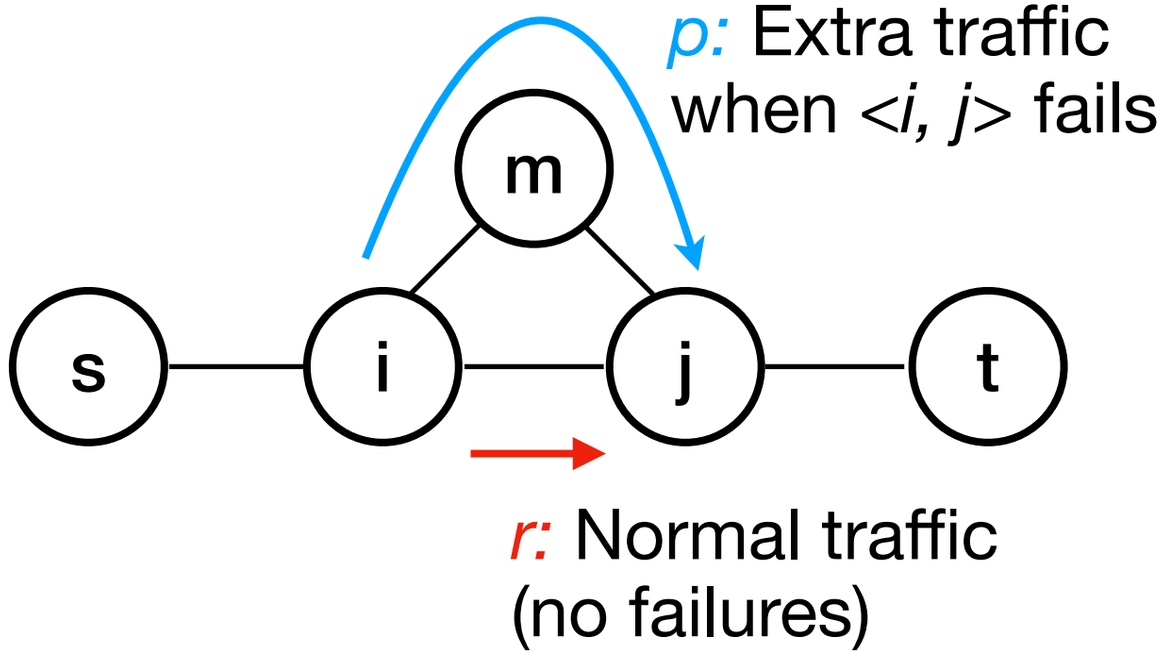


Lancet (top) vs. Gen-R3 (bottom)

on Abilene k = 1

Protection Routing Design

- Link-based protection routing
  - Provision by-pass paths to protect against each link failure
  - Achieved using (H), generalizing a state-of-the-art scheme [1]
- Issues with existing protection routing schemes
  - Only work if *X* is all f failures
  - If worst-case utilization > 1, we are forced to design for *f - 1* failures
- What we want: Design for most *f* failures if not all



$$\text{(H)} \quad \min_{r,p,a,U} \quad U$$

s.t. r_{st} is a unit flow from s to t. $\quad \forall s,t \in V$

p_l is a flow of a_l from i to j. $\quad \forall l \in E, l = \langle i,j \rangle$

$\forall x \in X, e \in E,$

$$\sum_{s,t} d_{st} r_{st}(e) + \sum_{l \in E} x_l p_l(e) \le U c_e (1 - x_e) + a_e x_e$$

$$a_e \ge 0 \quad \forall e \in E; \quad U \ge 0$$

[1] Wang, et al, R3: Resilient routing reconfiguration, Sigcomm 2010