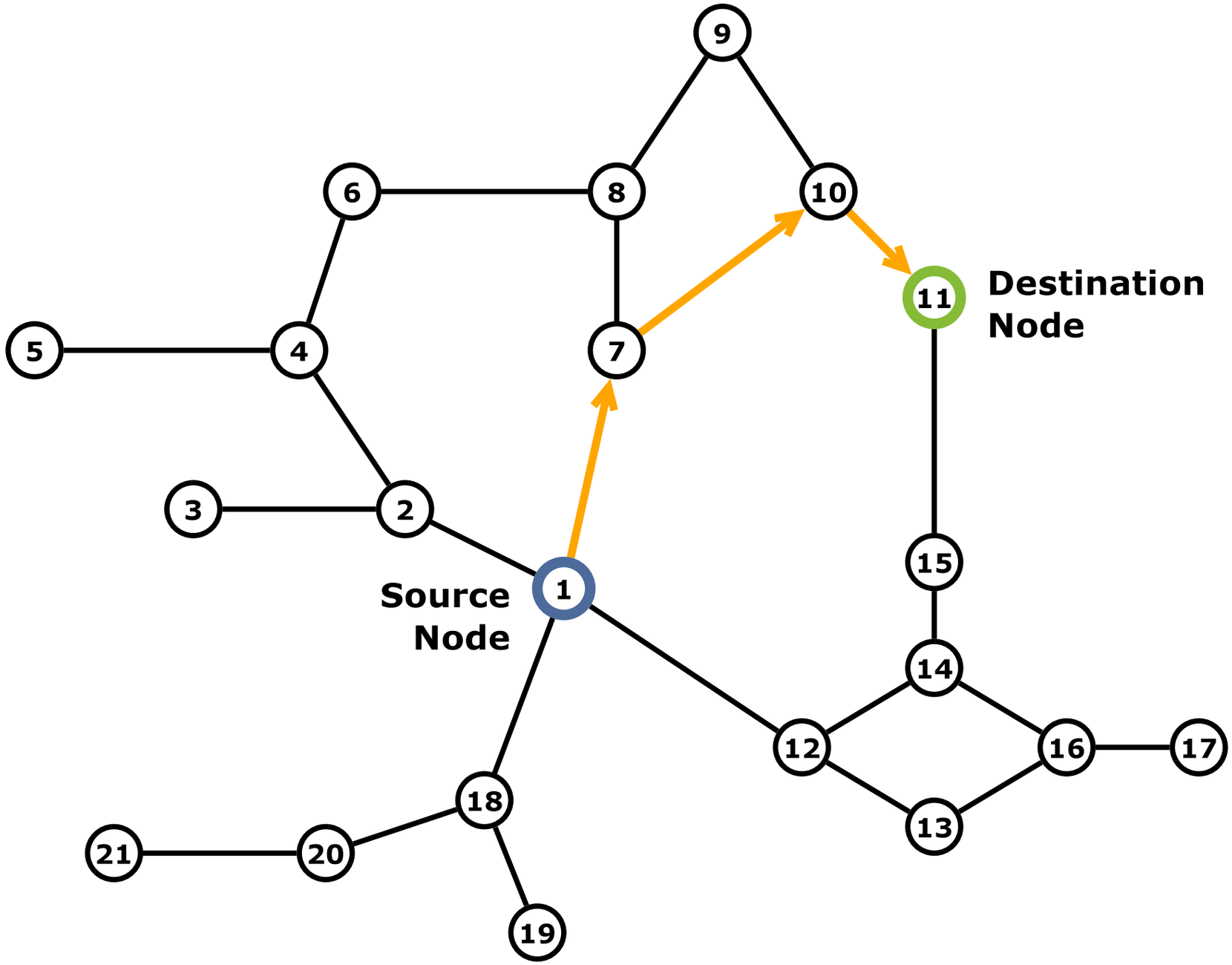
Bonsai: Efficient Fast Failover Routing Using Small Arborescences

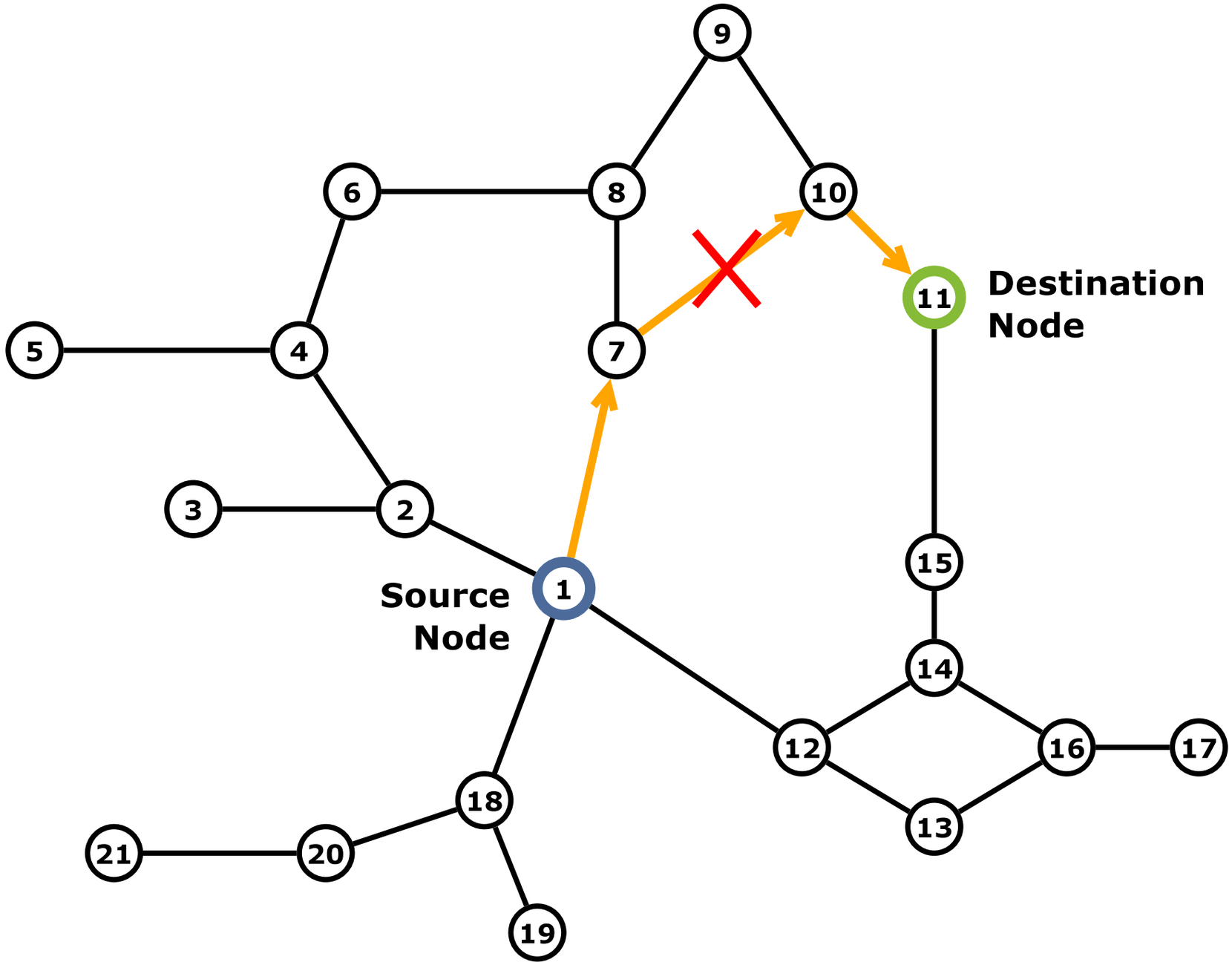
Klaus-Tycho Foerster (University of Vienna, Austria)
Andrzej Kamisiński (AGH University of Science and Technology in Kraków, Poland)
Yvonne-Anne Pignolet (DFINITY, Switzerland)
Stefan Schmid (University of Vienna, Austria)
Gilles Tredan (LAAS-CNRS, France)

DSN 2019, Portland, Oregon, June 26, 2019

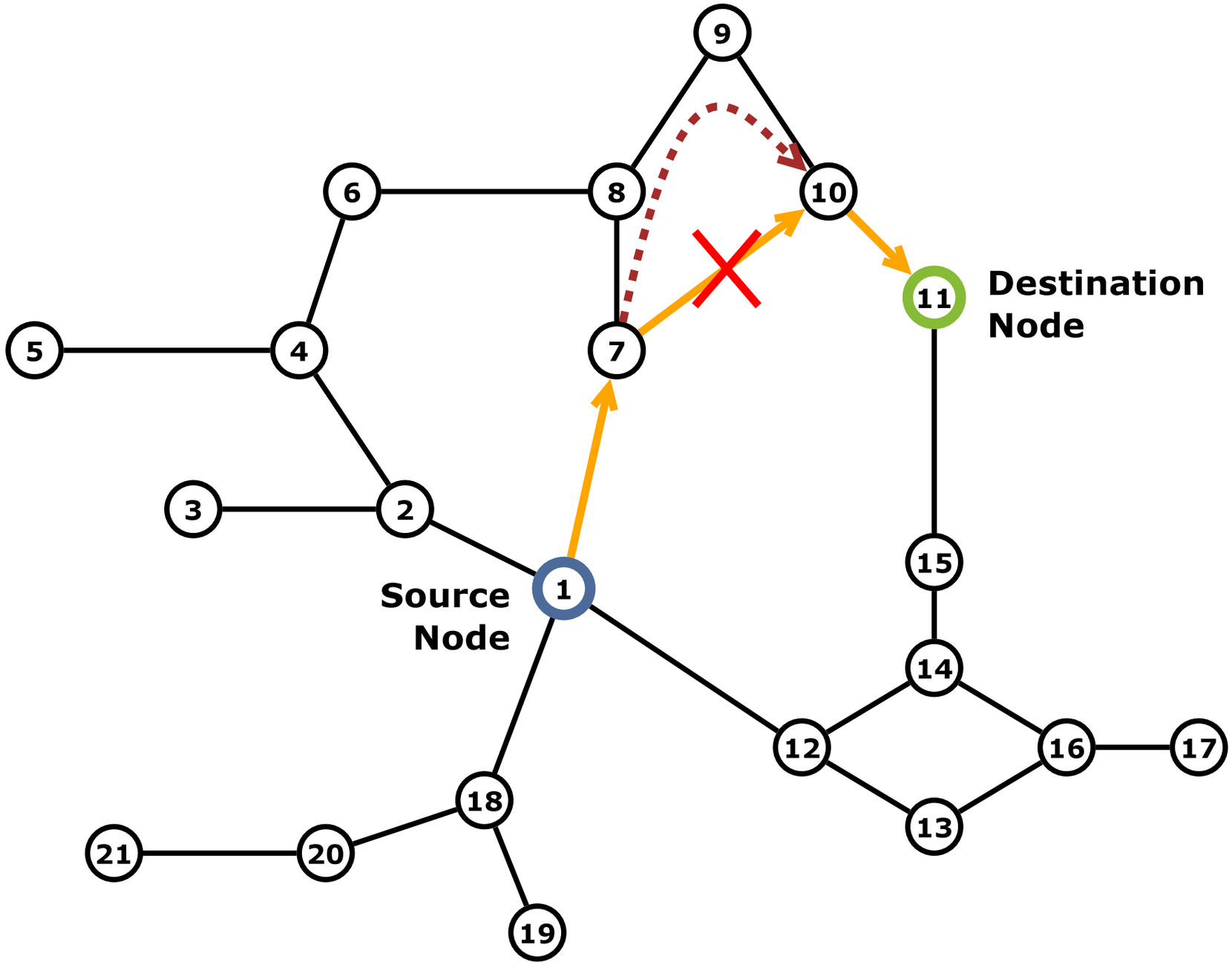
Introduction



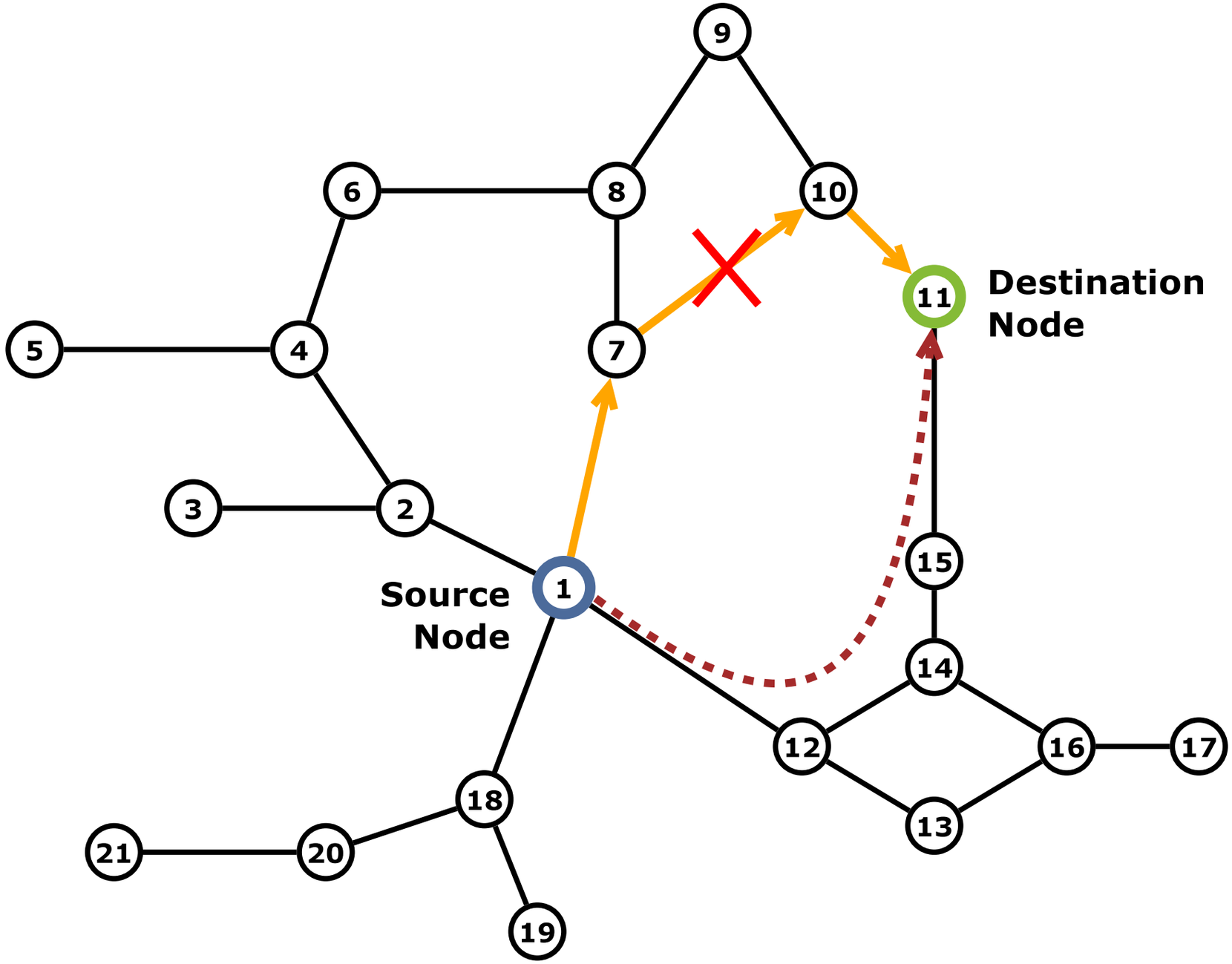
Introduction



Introduction



Introduction



How to deal with node/link failures?

Existing reconvergence mechanisms relying on the control-plane

- Dynamic exchange of information between nodes (signaling)
- Relatively long recovery time (can be shortened by appropriate tuning and fast detection mechanisms)
- Users may experience packet losses and increased delay

How to deal with node/link failures?

Existing reconvergence mechanisms relying on the control-plane

- Dynamic exchange of information between nodes (signaling)
- Relatively long recovery time (can be shortened by appropriate tuning and fast detection mechanisms)
- Users may experience packet losses and increased delay

Fast-Reroute mechanisms

- Local decision, limited delay and packet losses
- Forwarding to a loop-free alternate
- Input interface-aware routing
- Additional/extended forwarding tables and other data structures
- Tunneling
- Redundant spanning trees
- Failure coverage analysis and improvements (e.g., network graph modifications, virtual overlays, optimization of link costs, loop detection extensions)

How to deal with node/link failures?

Existing reconvergence mechanisms relying on the control-plane

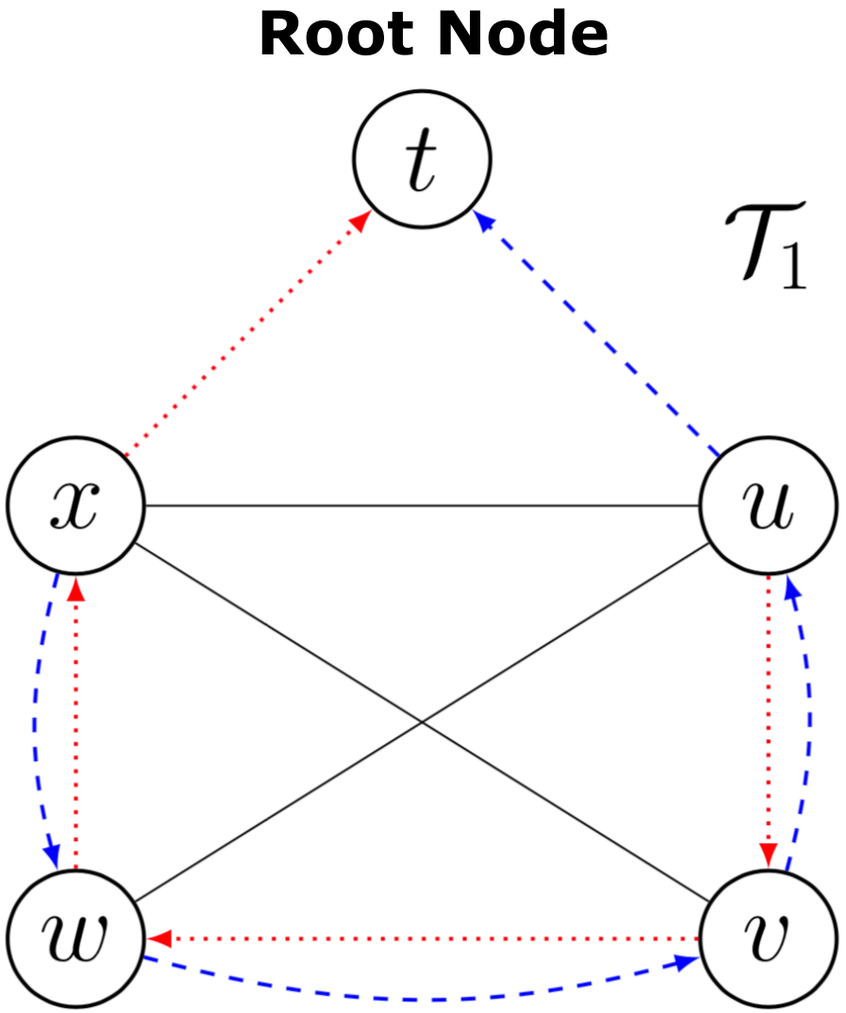
- Dynamic exchange of information between nodes (signaling)
- Relatively long recovery time (can be shortened by appropriate tuning and fast detection mechanisms)
- Users may experience packet losses and increased delay

Fast-Reroute mechanisms

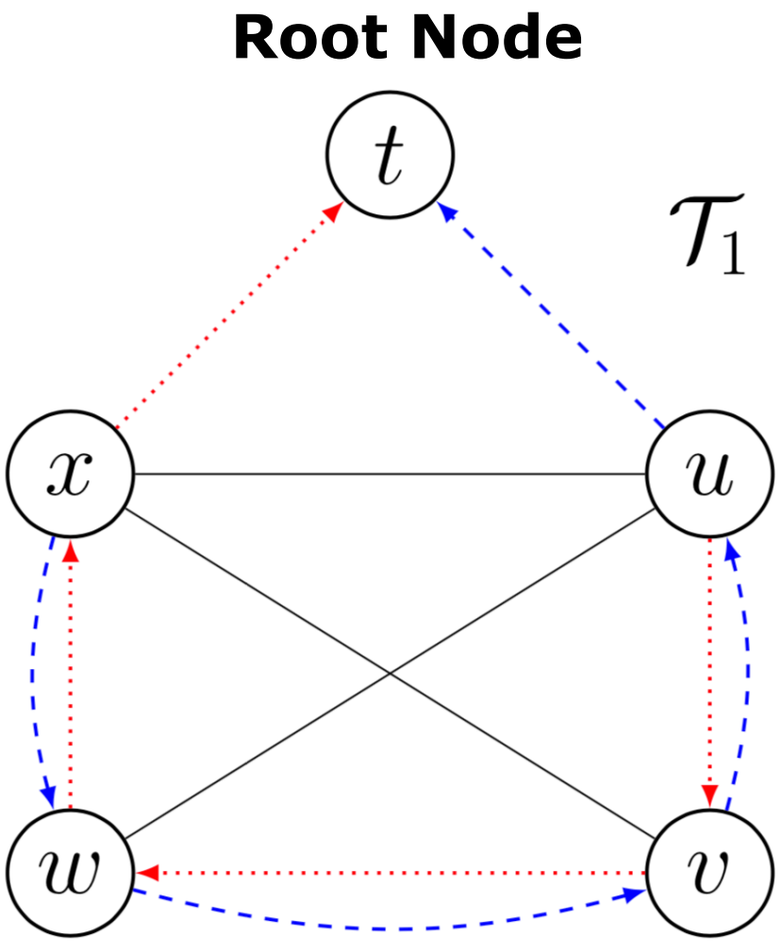
- Local decision, limited delay and packet losses
- Forwarding to a loop-free alternate
- Input interface-aware routing
- Additional/extended forwarding tables and other data structures
- Tunneling
- **Redundant spanning trees [THIS TALK]**
- Failure coverage analysis and improvements (e.g., network graph modifications, virtual overlays, optimization of link costs, loop detection extensions)

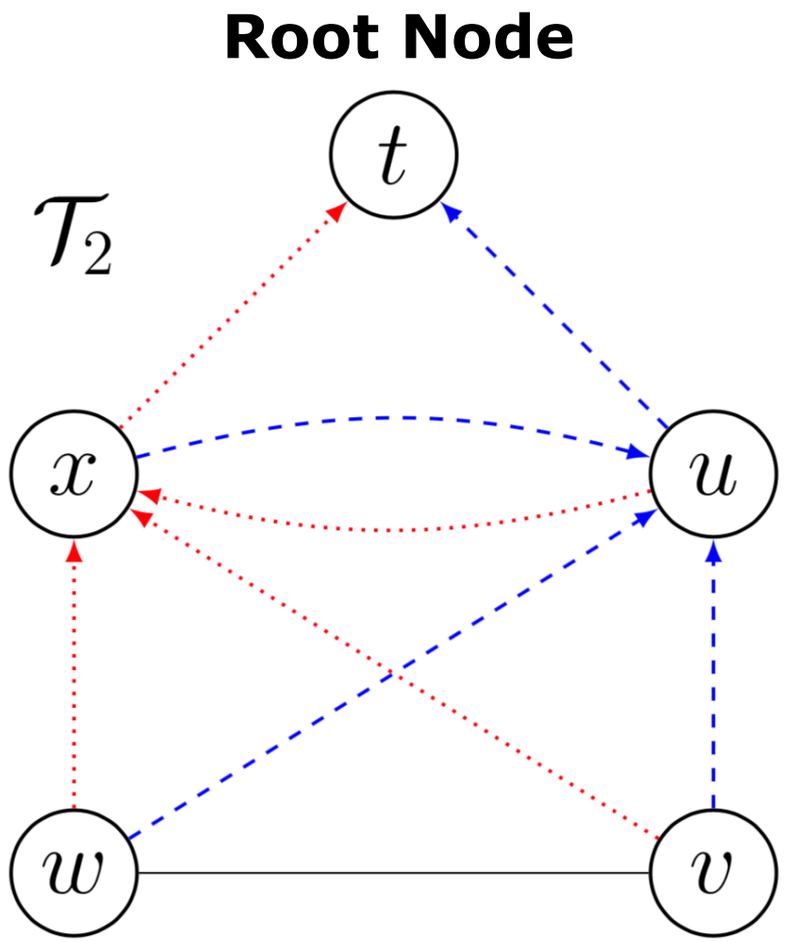
Motivation

Arborescence — a rooted directed spanning tree



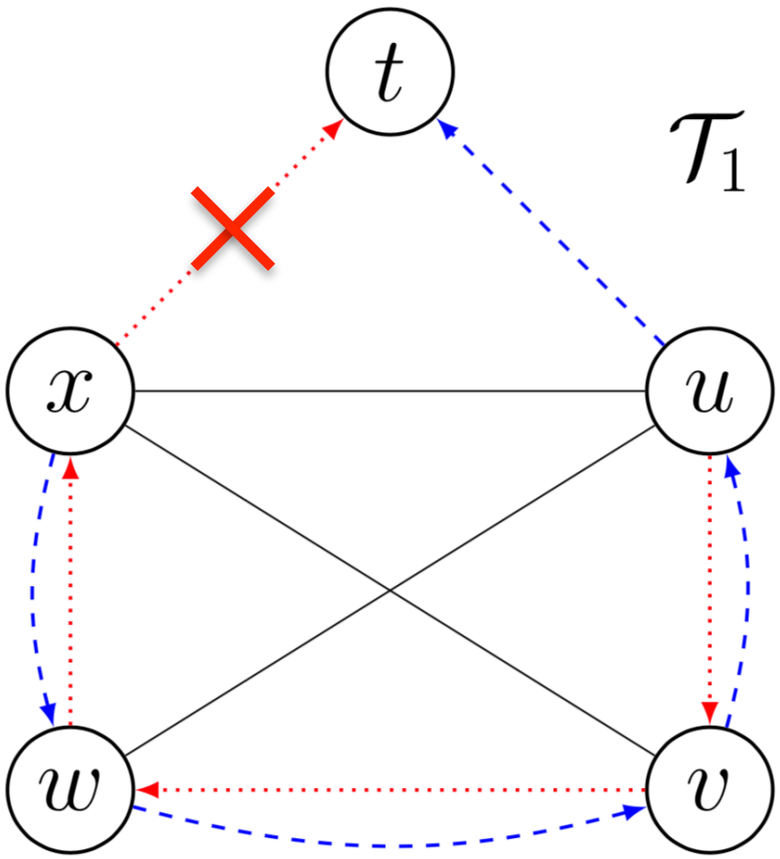
Motivation





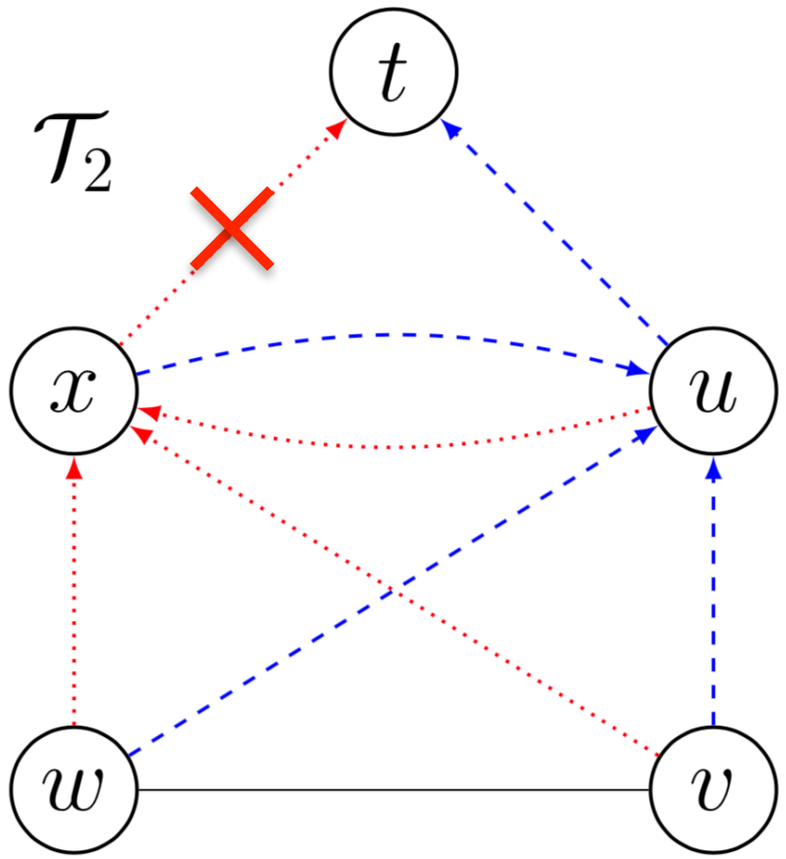
Motivation

Root Node



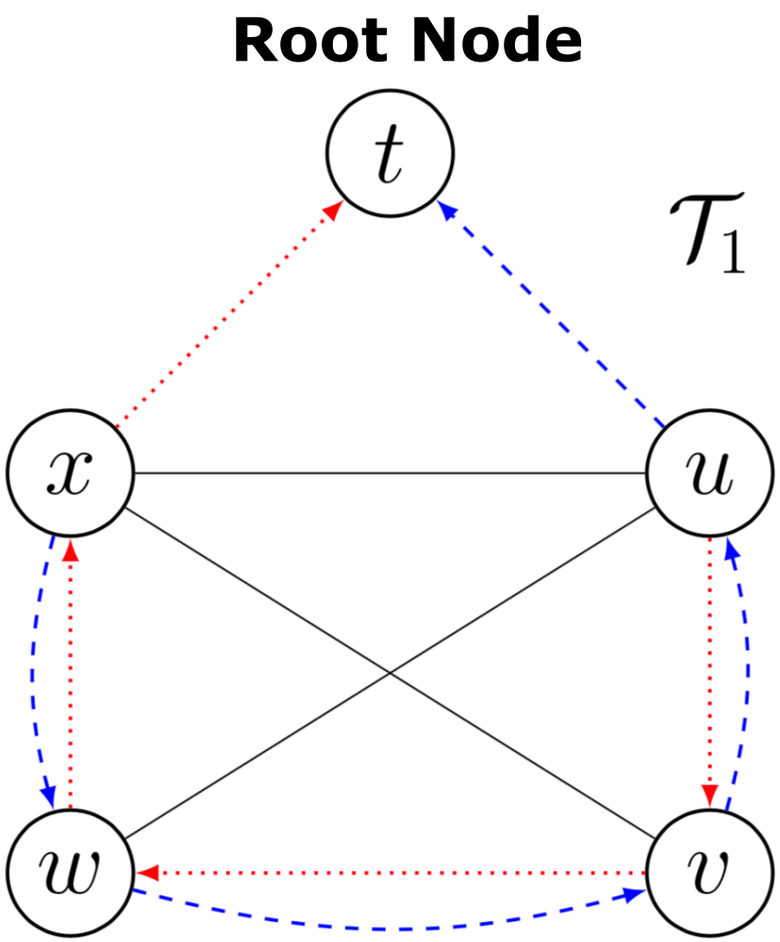
From *x* to *t*: 4 links

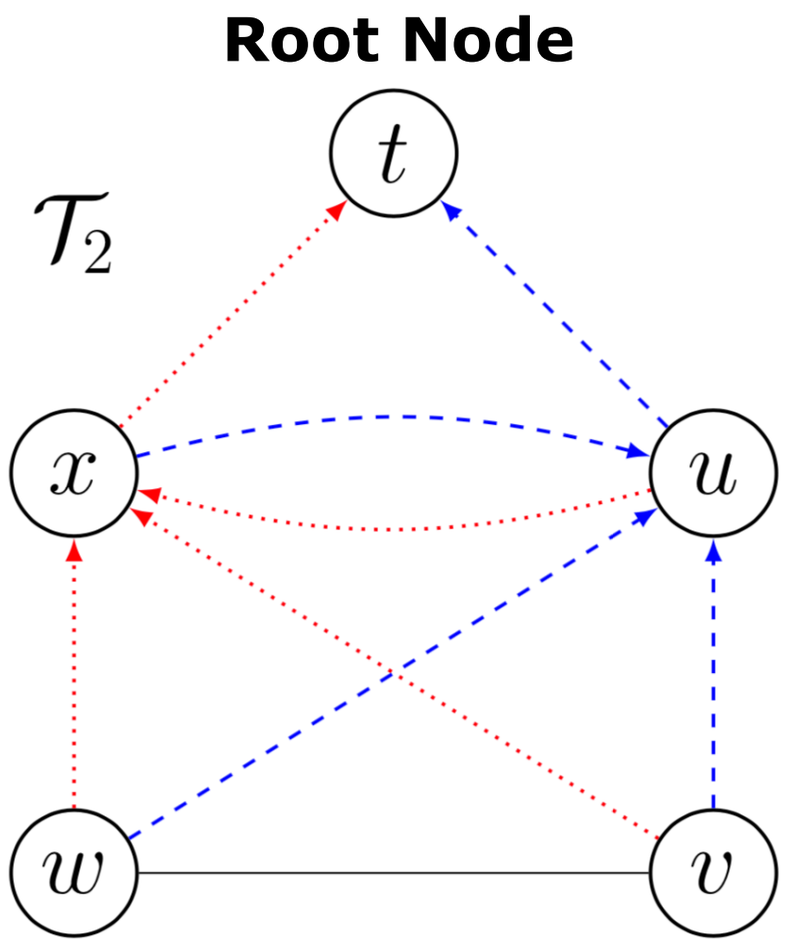
Root Node



From *x* to *t*: 2 links

Motivation





The way in which arborescences are constructed influences the length of alternative paths (delay, resource utilization).

Motivation

Existing algorithms based on redundant arborescences

- Focused mostly on network connectivity alone
- The resulting routes after failures can be very long, which in turn can harm performance
- Often require dynamic routing tables or modifications of packet headers
- Do not guarantee any non-trivial deterministic bounds on the resulting path lengths

Bonsai: How to Build Better Arborescences for Arbitrary Networks?

- **Arc-disjoint spanning arborescences** defining alternative routes to the destination

- We aim to keep the **path stretch** imposed by these trees **low** (hence the name of our method: *Bonsai*), **without sacrificing connectivity**

- **Guaranteed connectivity** in *k*-connected graphs even under many concurrent link failures, **independent of any dynamic state** at nodes

- The underlying problem is **NP-hard** on general network topologies (see the paper for details)
  - We present the lower bounds for various network graphs
  - We propose heuristics for general networks (Round-Robin Approach, Round-Robin with Link Swapping) and compare them with the state-of-the-art solution

https://www.bol.com

Outline

- Related algorithms and the proposed heuristics
- The selected evaluation results
- Conclusion

Random Decomposition

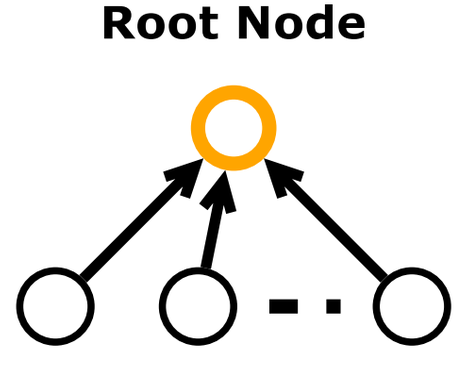
- Start at the root node

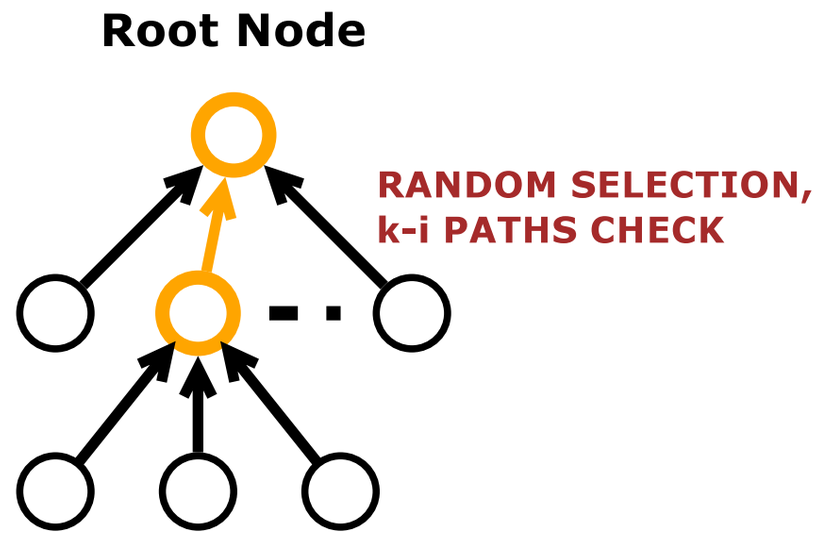
- Insert a random **unused** neighbor arc towards the root into arborescence T_i

- Extend T_i by adding more arcs recursively, maintaining the appropriate structure of the arborescence

- CHECK IF EXIST: ***k−i* arc-disjoint paths** from the considered neighbor to the root

- This algorithm **always succeeds** in constructing *k* arc-disjoint arborescences





State of the Art: Greedy Decomposition (Chiesa et al.)

- Start at the root node

- Insert one of the **unused** neighbor arcs towards the root into arborescence T_i and Consider all candidate neighbor arcs ordered by **increasing depth** of arborescences

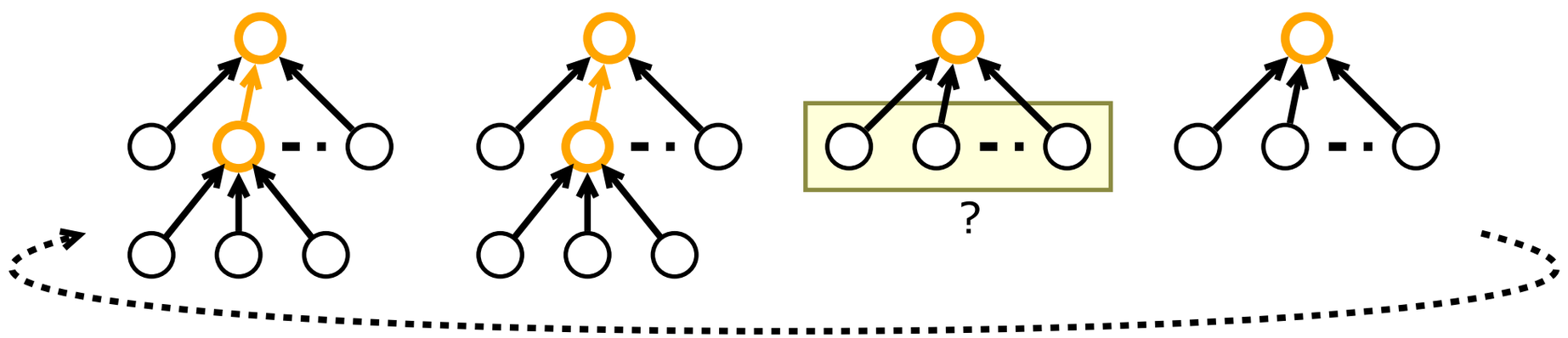
- CHECK IF EXIST: $k-i$ arc-disjoint paths from the considered neighbor to the root

- The **depth** of the first arborescence is the **smallest** of all arborescences and the depth of the other arborescences increases monotonically

- This algorithm **always succeeds** in constructing k arc-disjoint arborescences

Round-Robin Approach

- Constructs arborescences **in parallel**
- Adding one arc per arborescence in a Round-Robin fashion
- Increasing the **tree depth** only if strictly necessary
- Much more **balanced arborescence packings** with respect to the length of the detours they entail
- This procedure does **not always** succeed for general graphs (in some cases, there is no unused arc left that can be added to the current arborescence)



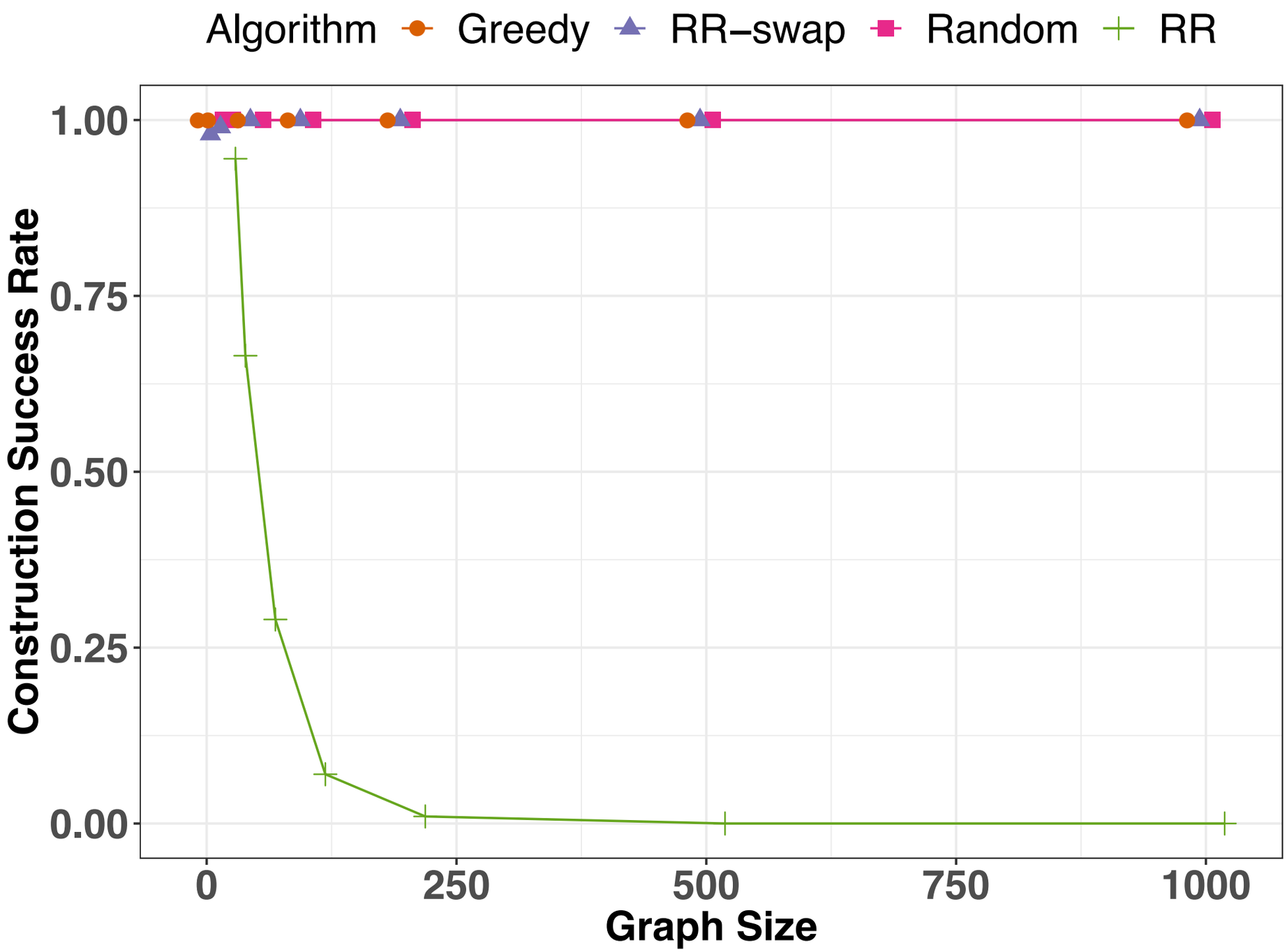
Swapping Arcs when Growing Arborescences

- When arc (u, v) cannot be added to arborescence T_i in the normal Round-Robin fashion, try to swap arcs
  $e = (u, v)$, $e' = (u, v')$

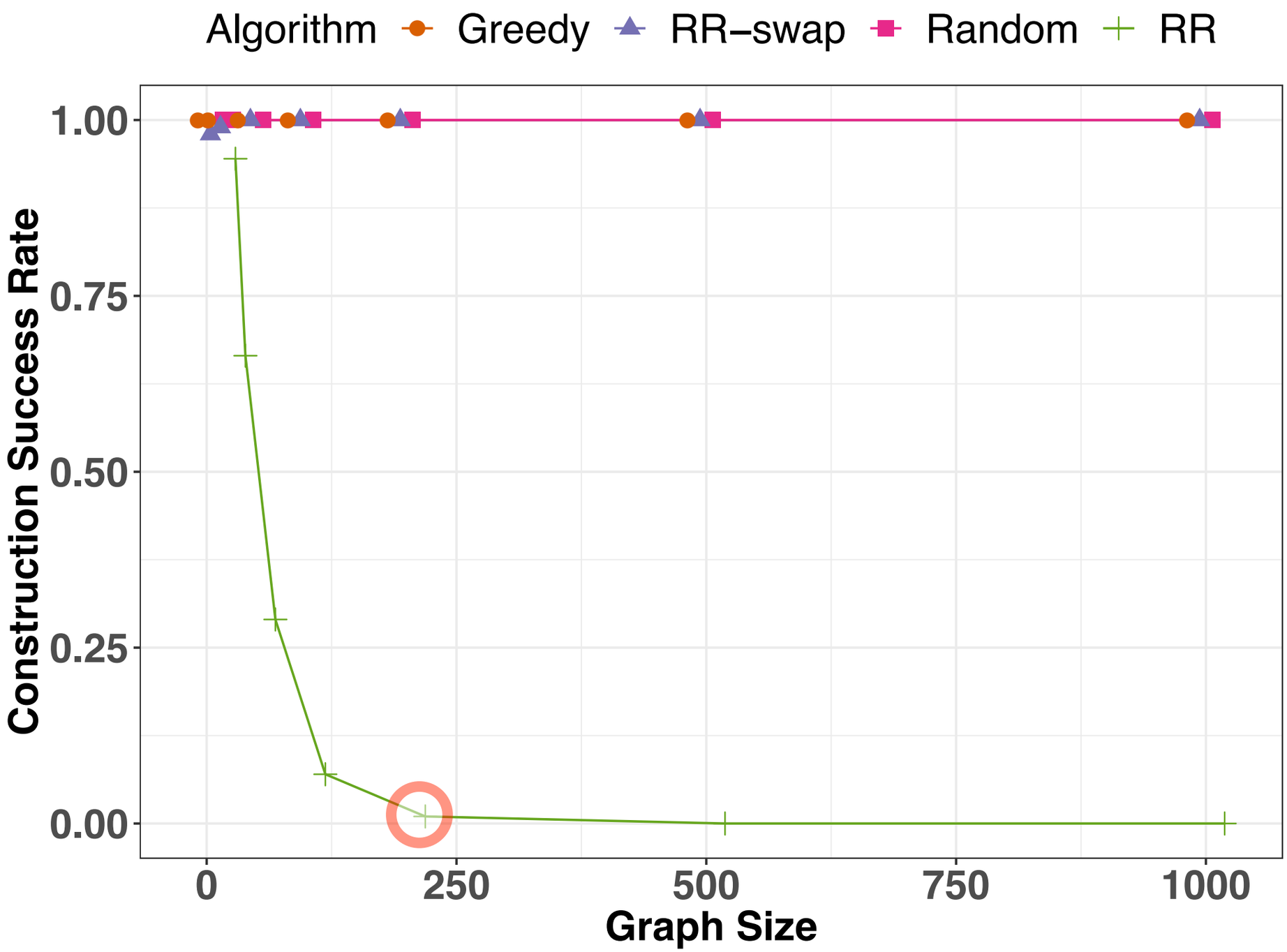
- Much more efficient than the naive approach which checks for all pairs of arcs if the involved graphs T_i , T_j are still valid arborescences after the swap

- If it fails, one may still fall back to the Greedy Decomposition

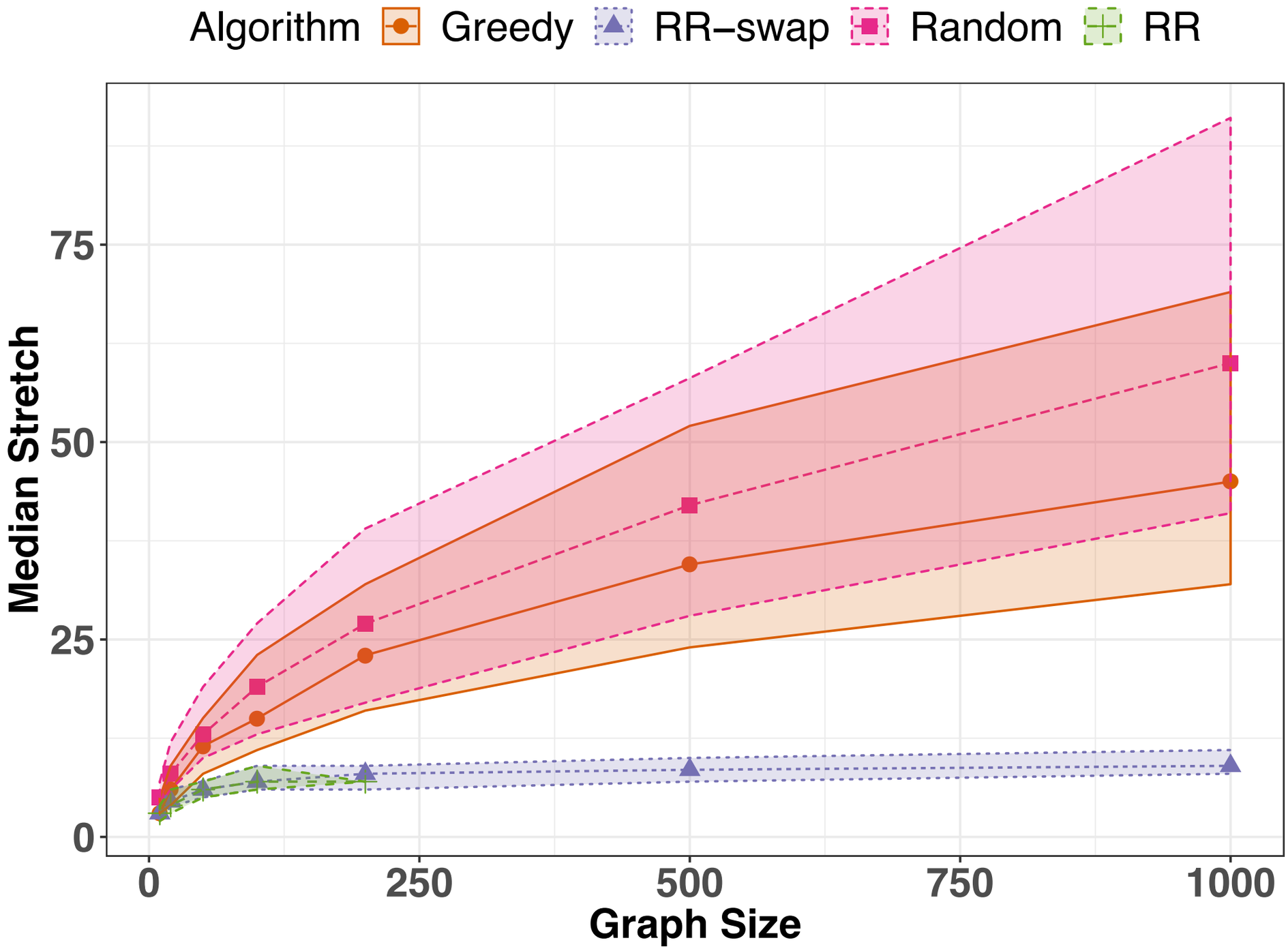
Evaluation Results: Construction Success Rate
Random *k*-regular Graphs



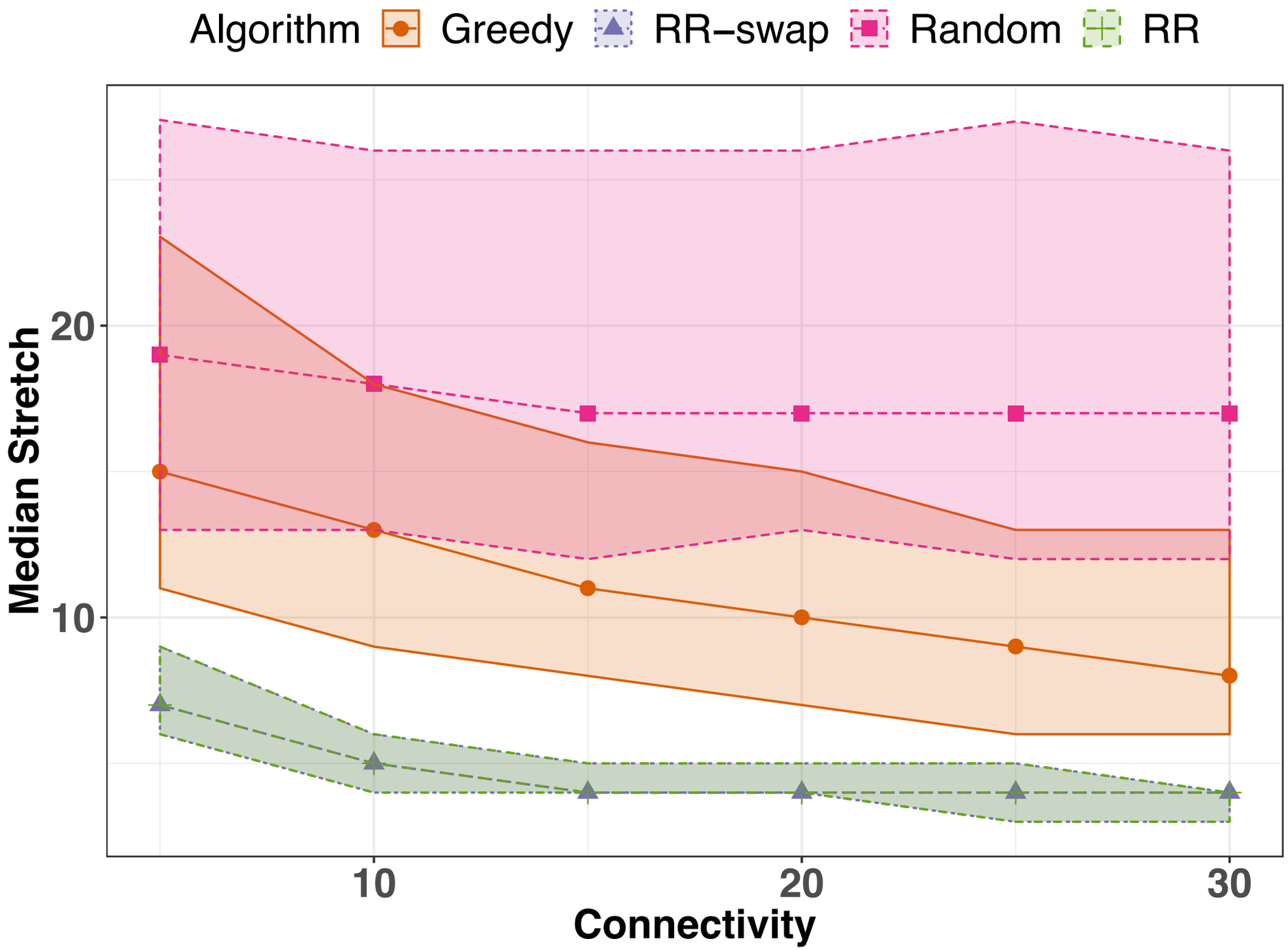
Evaluation Results: Construction Success Rate
Random *k*-regular Graphs



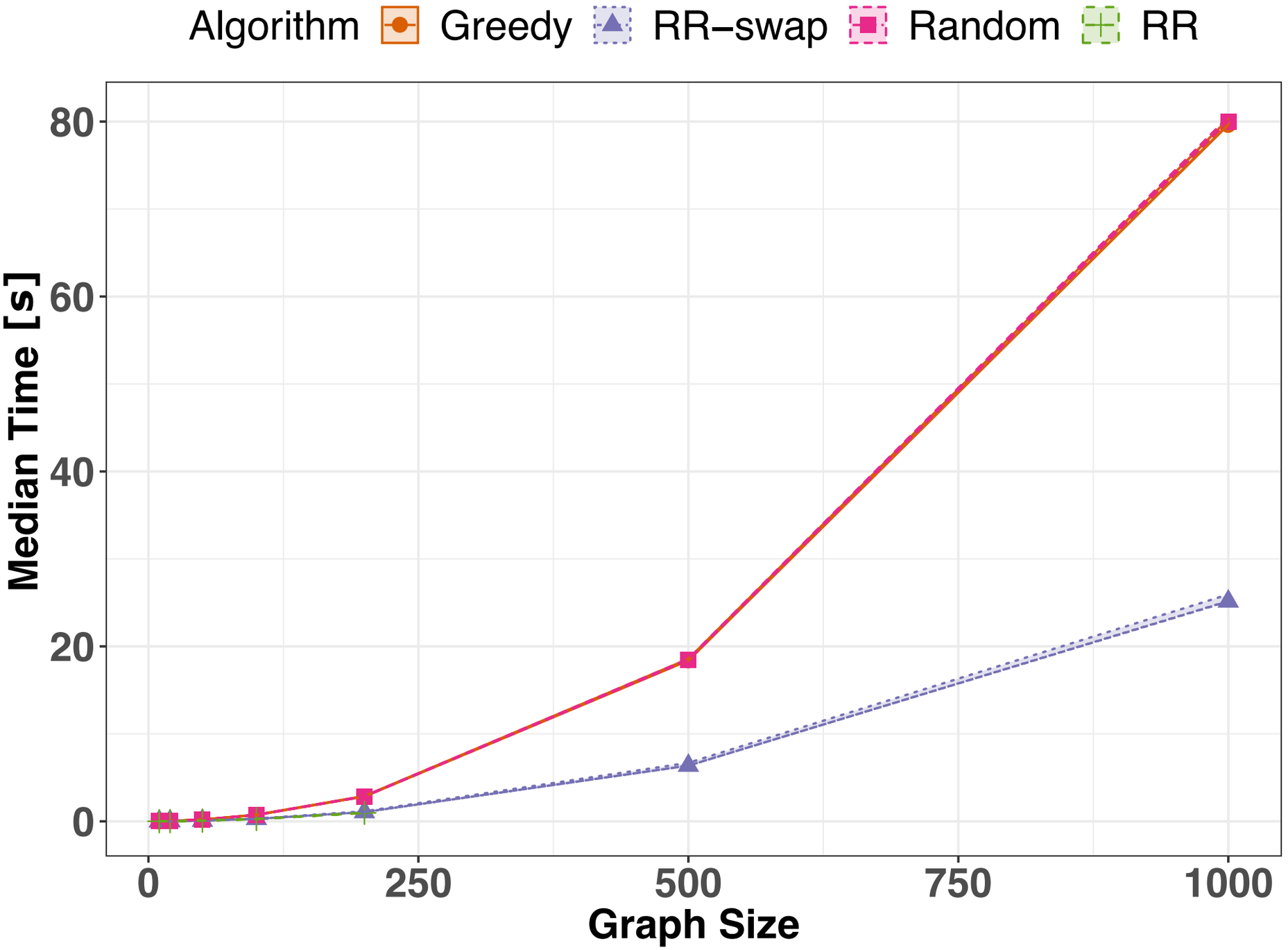
Evaluation Results: Median Path Stretch (1)
Random *k*-regular Graphs



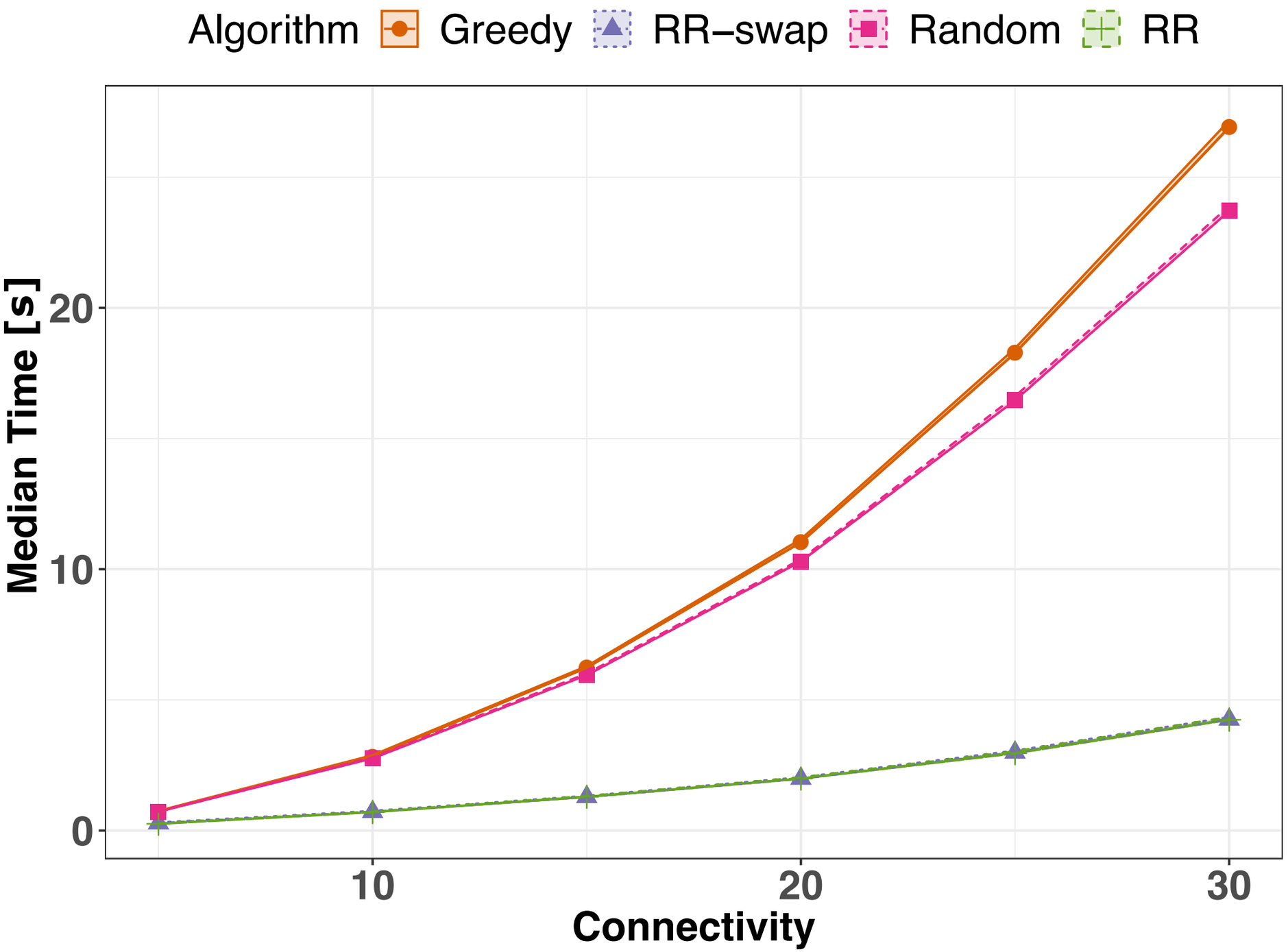
Evaluation Results: Median Path Stretch (2)
Random *k*-regular Graphs



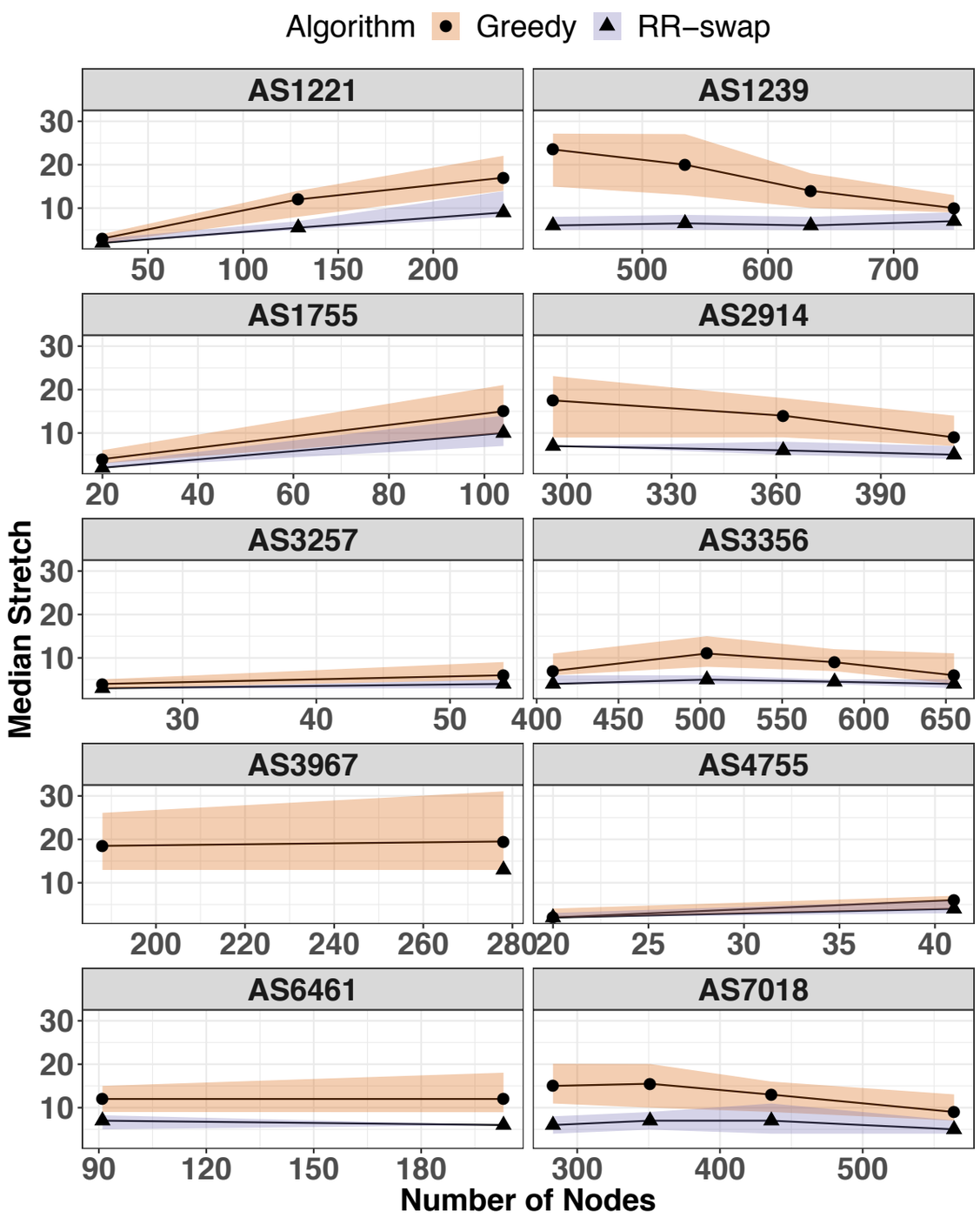
Evaluation Results: Median Computation Time (1)
Random *k*-regular Graphs



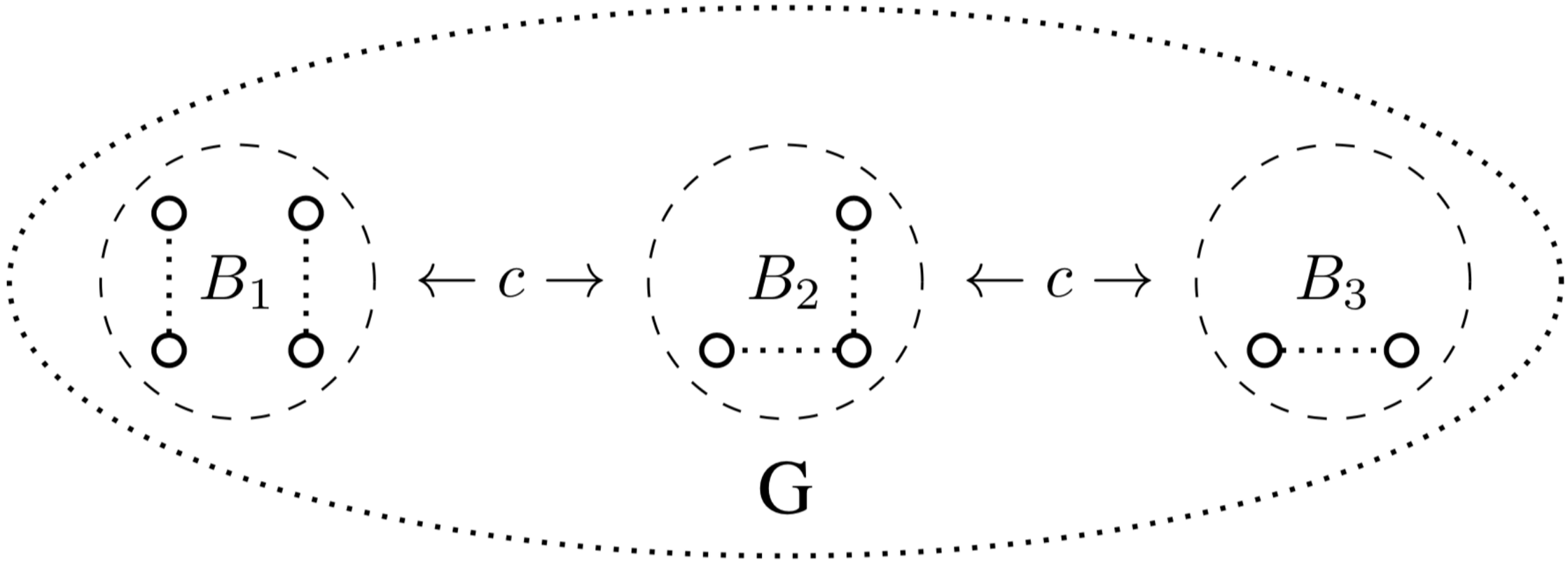
Evaluation Results: Median Computation Time (2)
Random *k*-regular Graphs



Evaluation Results: Median Path Stretch Well-connected Cores of Different AS Graphs



Failure Balls



Conclusion

- Heuristics supporting construction of arc-disjoint arborescences
  - Short failover routes with optimized path stretch
  - Provide connectivity under many concurrent link failures

- The simulation results demonstrate feasibility

- The source code and simulation results will soon be available at: https://gitlab.cs.univie.ac.at/ct-papers/2019-dsn

Thank you for your attention

Bonsai: Efficient Fast Failover Routing Using Small Arborescences

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)

Presenter: Andrzej Kamisiński (andrzejk@agh.edu.pl)