IEEE INFOCOM

Looking for the Maximum Independent Set: A New Perspective on the Stable Path Problem

Yichao Cheng[1], Ning Luo[1], Jingxuan Zhang[1, 2],
Timos Antonopoulos[1], Ruzica Piskac[1], Qiao Xiang[3, 1],

[1]Yale University, [2]Tongji University, [3]Xiamen University

05/12/2021, IEEE INFOCOM'21

Yale University

同濟大學 TONGJI UNIVERSITY

厦門大學 XIAMEN UNIVERSITY

What is the Stable Path Problem?

Powerful tool for BGP convergence analysis [1]

[1] Griffin, Timothy, et al. "The Stable Paths Problem and Interdomain Routing." *IEEE/ACM Transactions On Networking* 10, no. 2 (2002): 232-243.

What is the Stable Path Problem?

Powerful tool for BGP convergence analysis [1]

- An undirected graph G=(V, E)
  - n+1 nodes, node 0 represents the destination; a link represents neighborship
- Each node i
  - runs SPVP, an abstract version of BGP
  - has a set of permitted paths
  - has a ranking function to rank permitted paths

[1] Griffin, Timothy, et al. "The Stable Paths Problem and Interdomain Routing." *IEEE/ACM Transactions On Networking* 10, no. 2 (2002): 232-243.

What is the Stable Path Problem?

Powerful tool for BGP convergence analysis [1]

- An undirected graph G=(V, E)
  - n+1 nodes, node 0 represents the destination; a link represents neighborship
- Each node i
  - runs SPVP, an abstract version of BGP
  - has a set of permitted paths
  - has a ranking function to rank permitted paths
- **Stable path assignment**: each node's selected path
  - belongs to its permitted paths, and
  - is a concatenation of the selected path of its neighbors

[1] Griffin, Timothy, et al. "The Stable Paths Problem and Interdomain Routing." *IEEE/ACM Transactions On Networking* 10, no. 2 (2002): 232-243.

What is the Stable Path Problem?

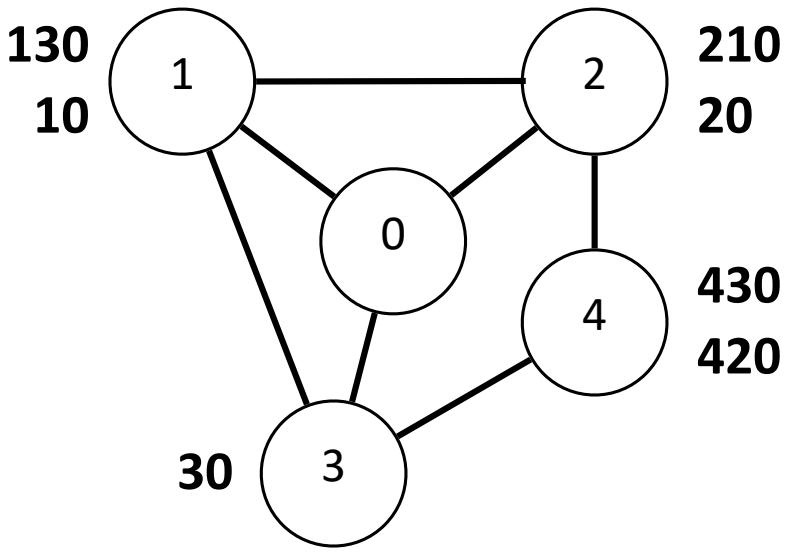
Powerful tool for BGP convergence analysis [1]

- An undirected graph G=(V, E)
  - n+1 nodes, node 0 represents the destination; a link represents neighborship
- Each node i
  - runs SPVP, an abstract version of BGP
  - has a set of permitted paths
  - has a ranking function to rank permitted paths
- **Stable path assignment**: each node's selected path
  - belongs to its permitted paths, and
  - is a concatenation of the selected path of its neighbors
- **Stable path problem (SPP)**: does a stable path assignment exists?

[1] Griffin, Timothy, et al. "The Stable Paths Problem and Interdomain Routing." *IEEE/ACM Transactions On Networking* 10, no. 2 (2002): 232-243.

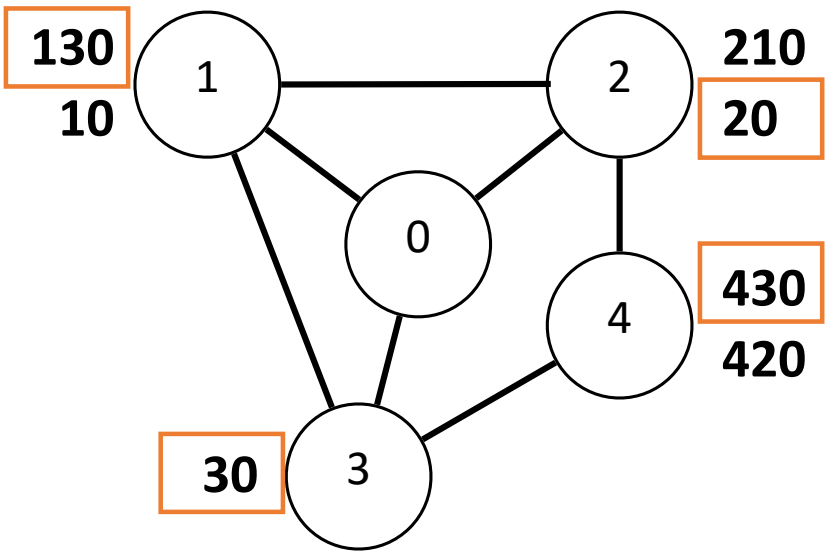
Example: Good Gadget

Example: Good Gadget



Good gadget

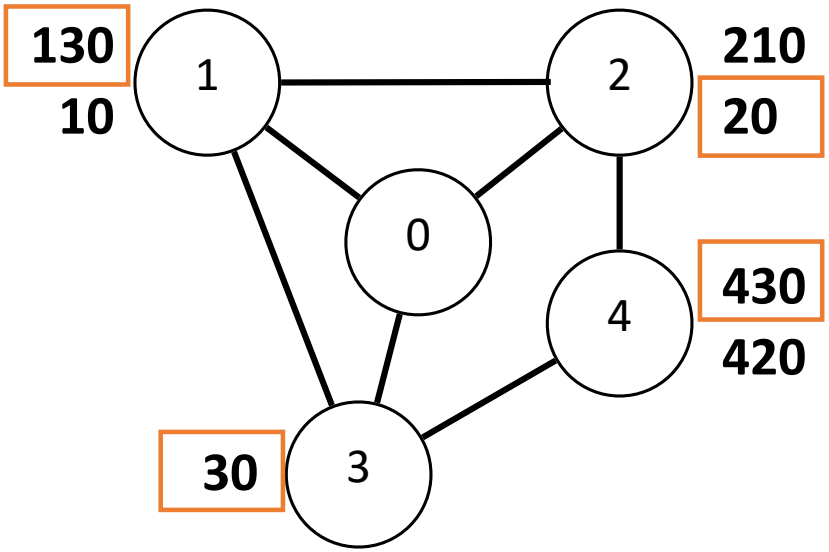
Example: Good Gadget



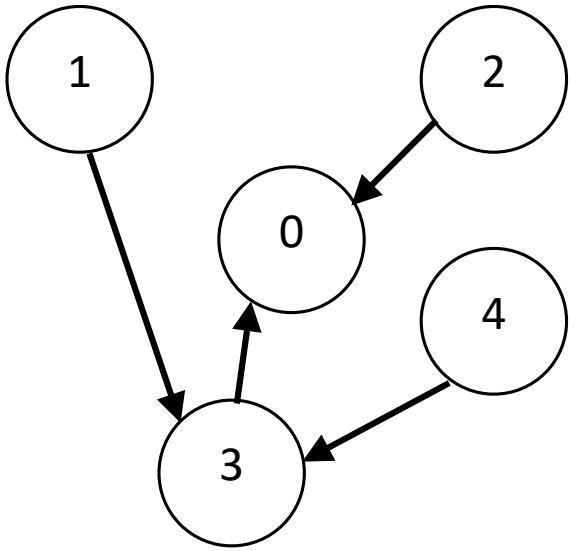
Good gadget

Example: Good Gadget

- A stable path assignment defines a tree rooted at node 0



Good gadget



Stable path assignment of good gadget

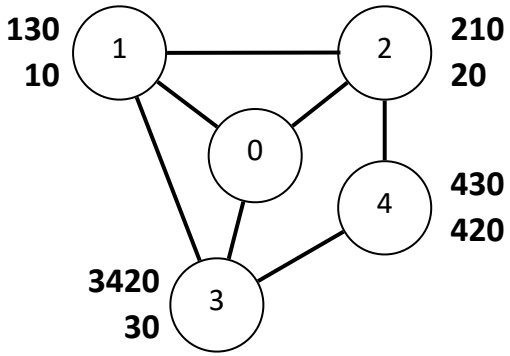
Categories on SPP Instances

Categories on SPP Instances

- **Safe**: SPVP always converges, e.g., good gadget

Categories on SPP Instances

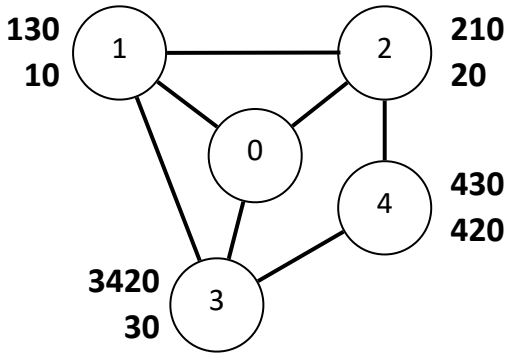
- **Safe**: SPVP always converges, e.g., good gadget
- **Unique**: has one and only one stable path assignment, e.g., good gadget and naughty gadget



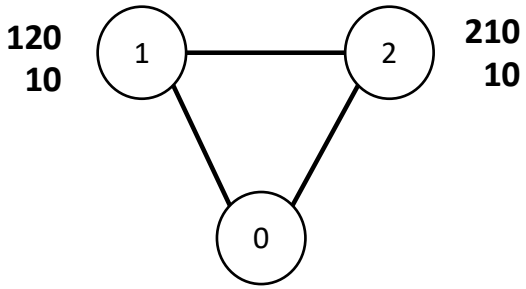
naughty gadget

Categories on SPP Instances

- **Safe**: SPVP always converges, e.g., good gadget
- **Unique**: has one and only one stable path assignment, e.g., good gadget and naughty gadget
- **Solvable**: has at least one stable path assignment, e.g., disagree



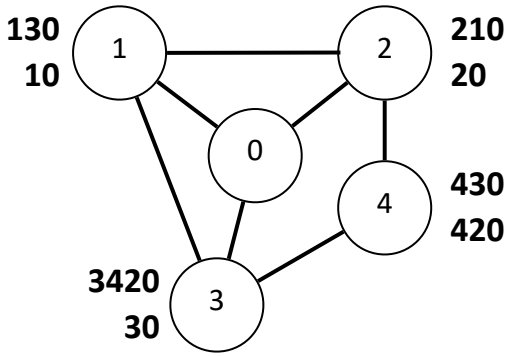
naughty gadget



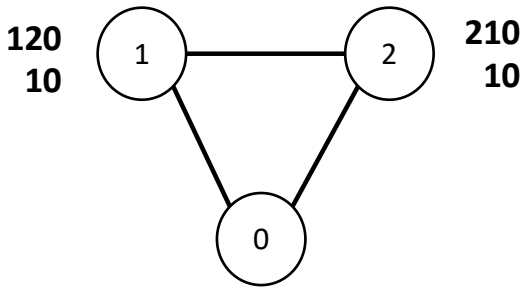
disagree

Categories on SPP Instances

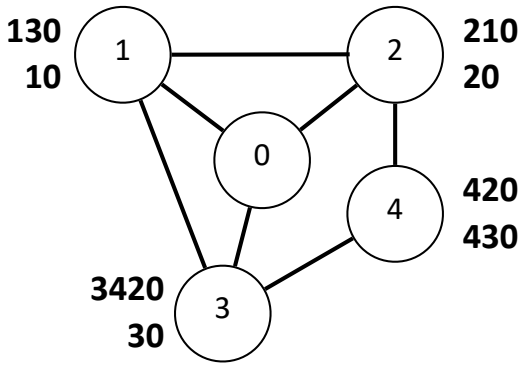
- **Safe**: SPVP always converges, e.g., good gadget
- **Unique**: has one and only one stable path assignment, e.g., good gadget and naughty gadget
- **Solvable**: has at least one stable path assignment, e.g., disagree
- **Unsolvable**: has no stable path assignment, e.g., bad gadget



naughty gadget



disagree

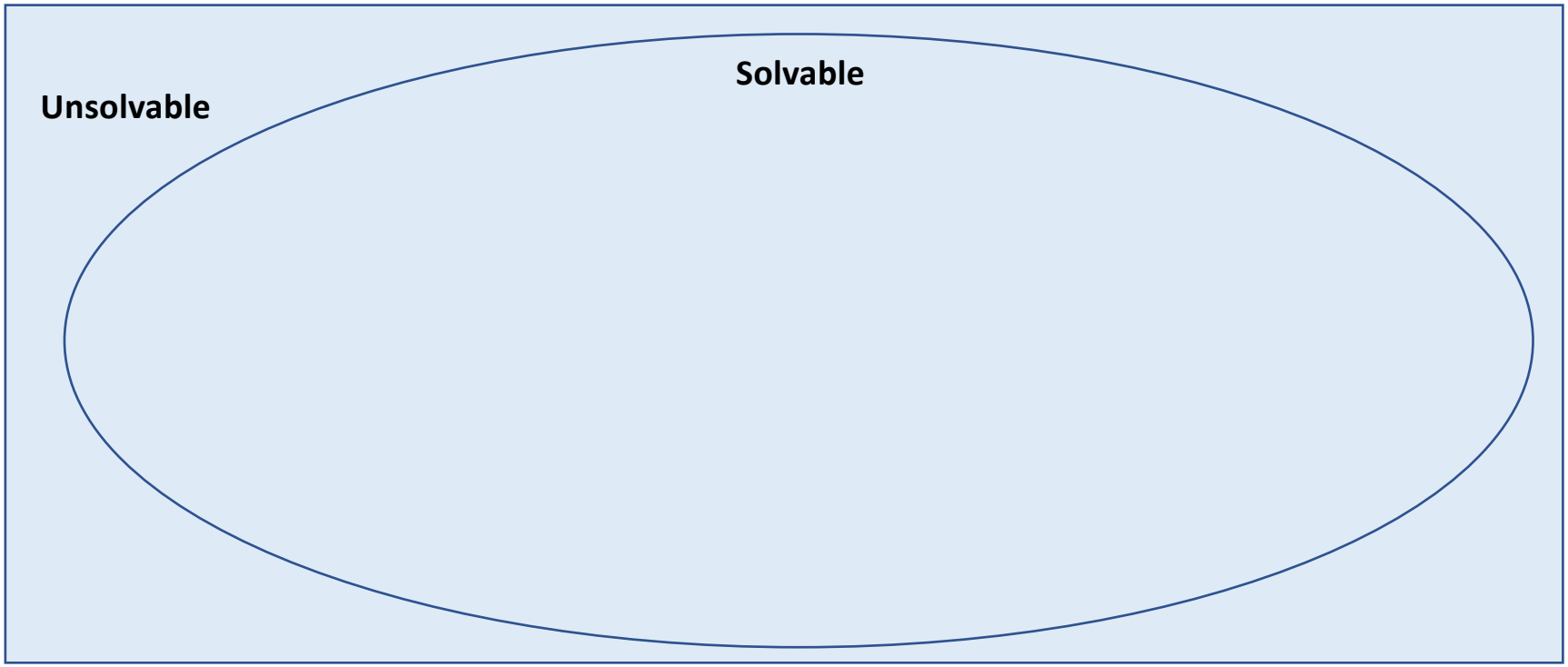


bad gadget

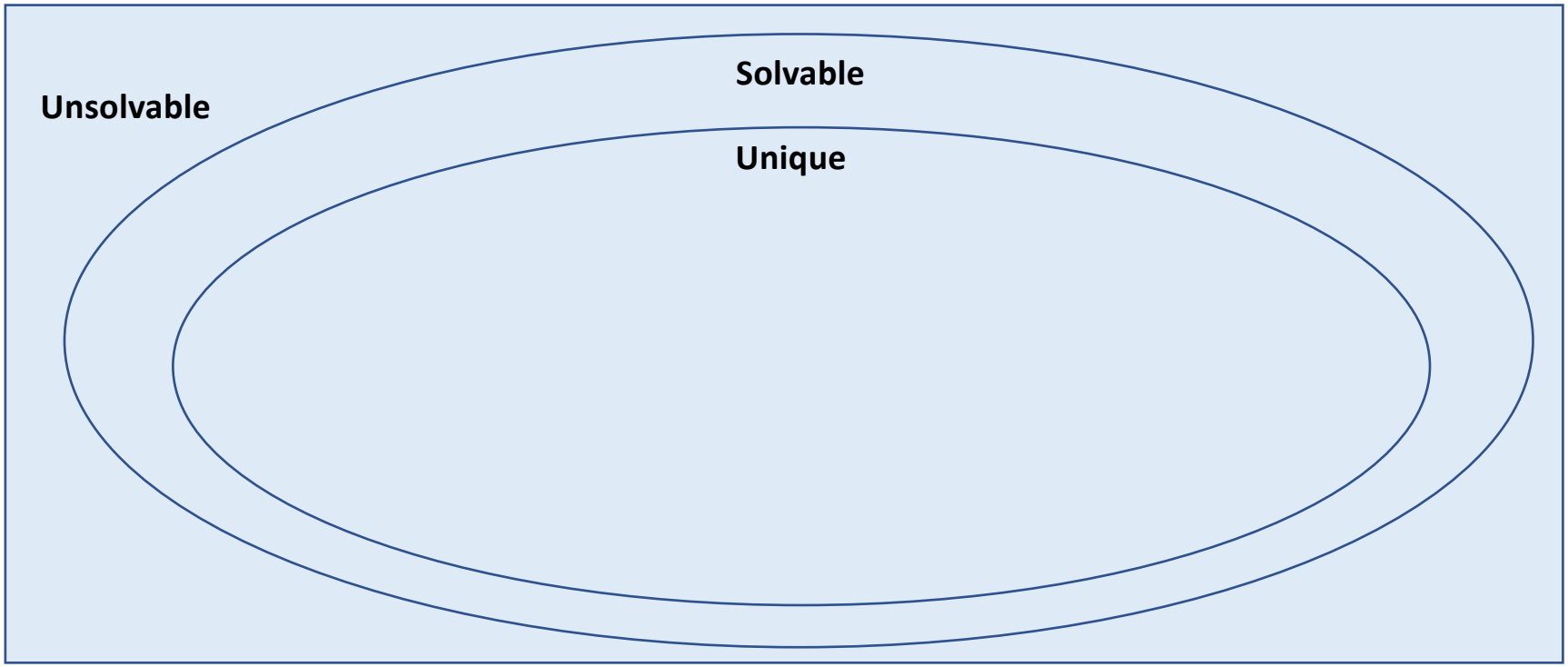
Related Work on SPP

Related Work on SPP

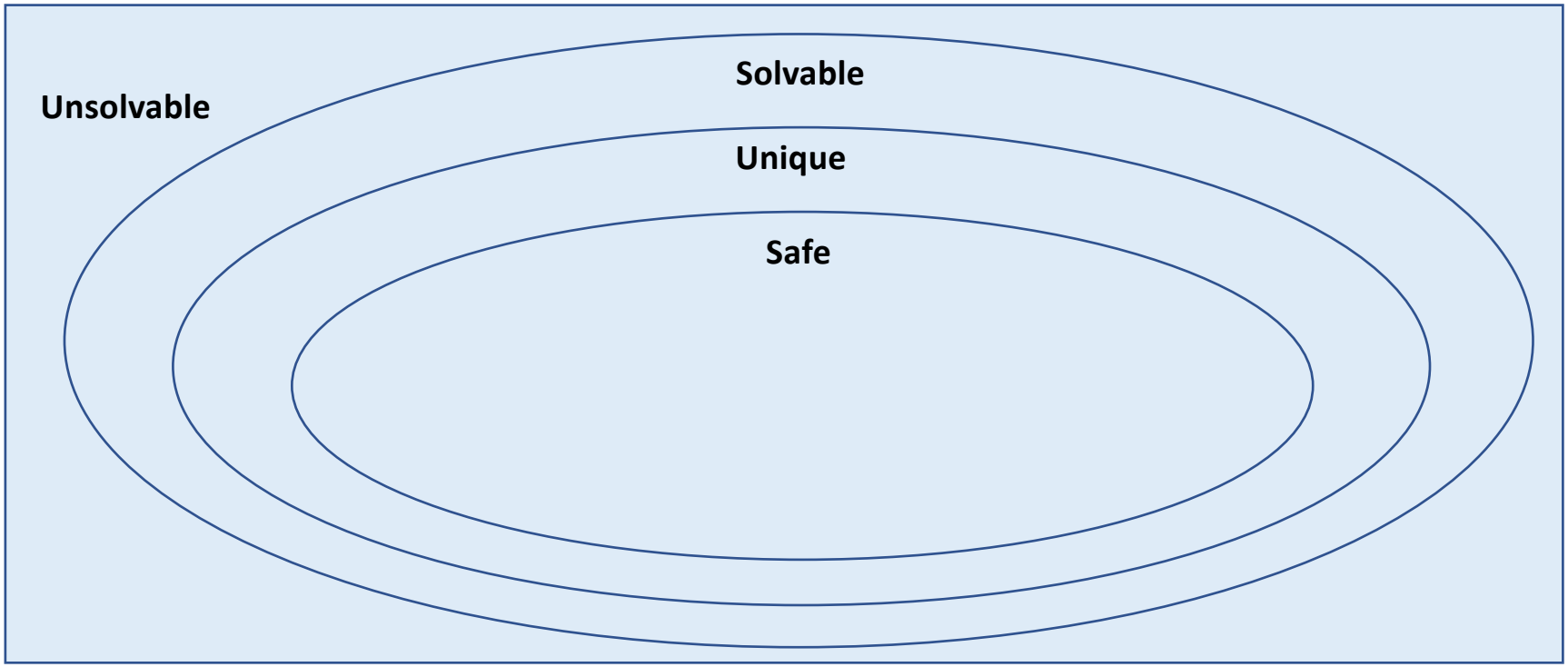
Related Work on SPP



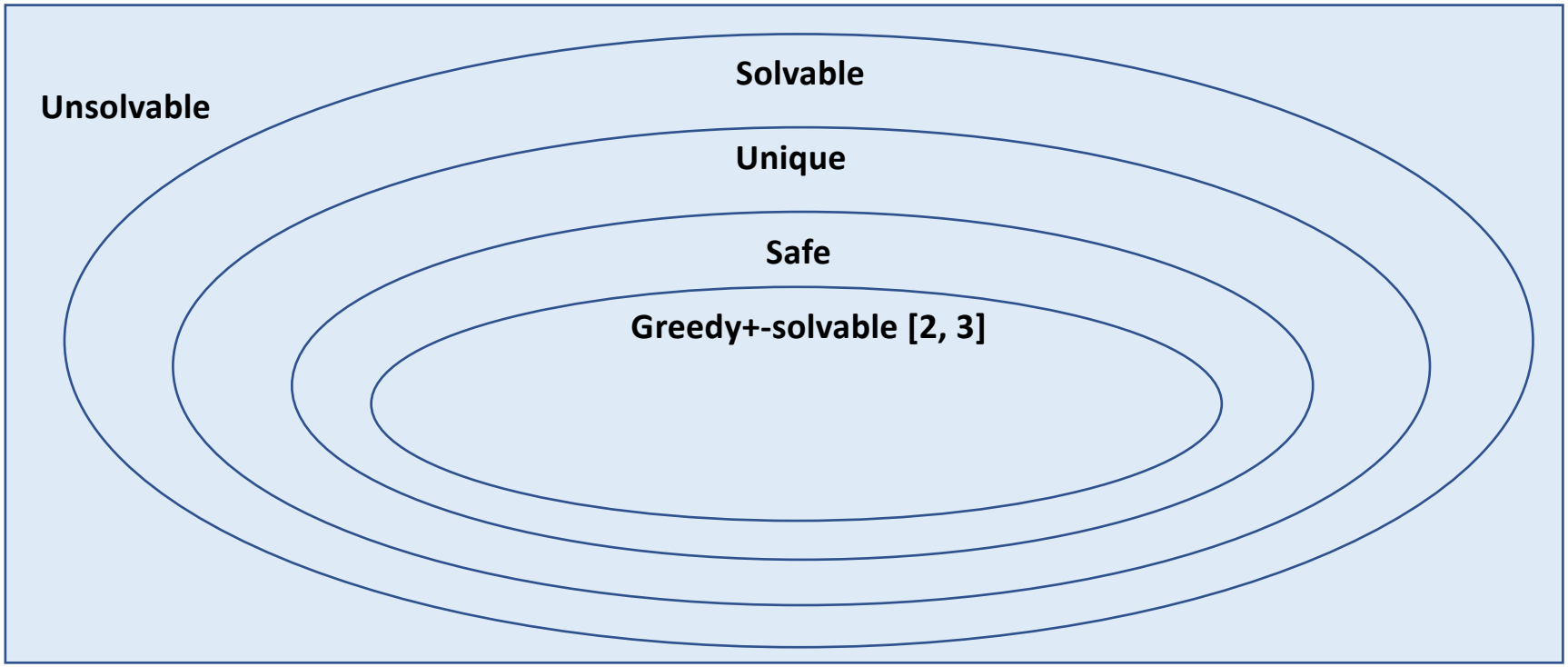
Related Work on SPP



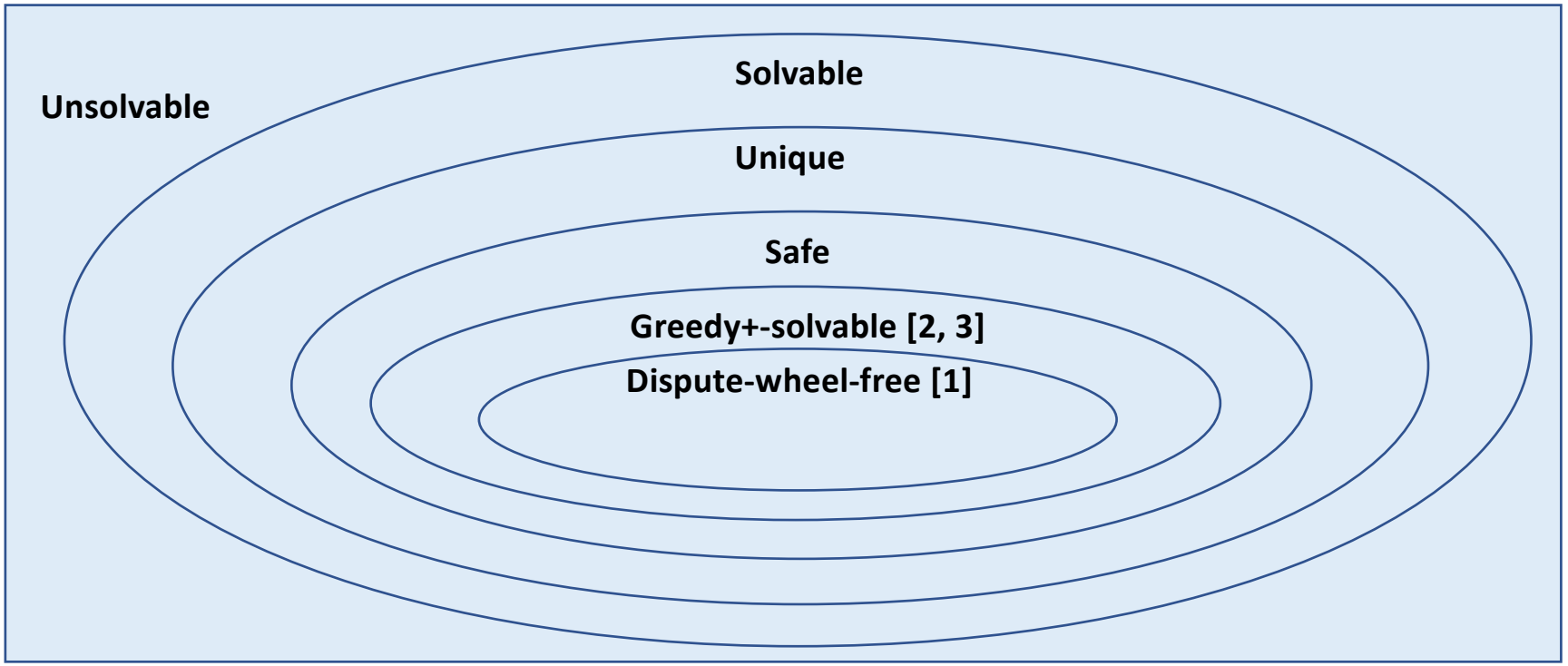
Related Work on SPP



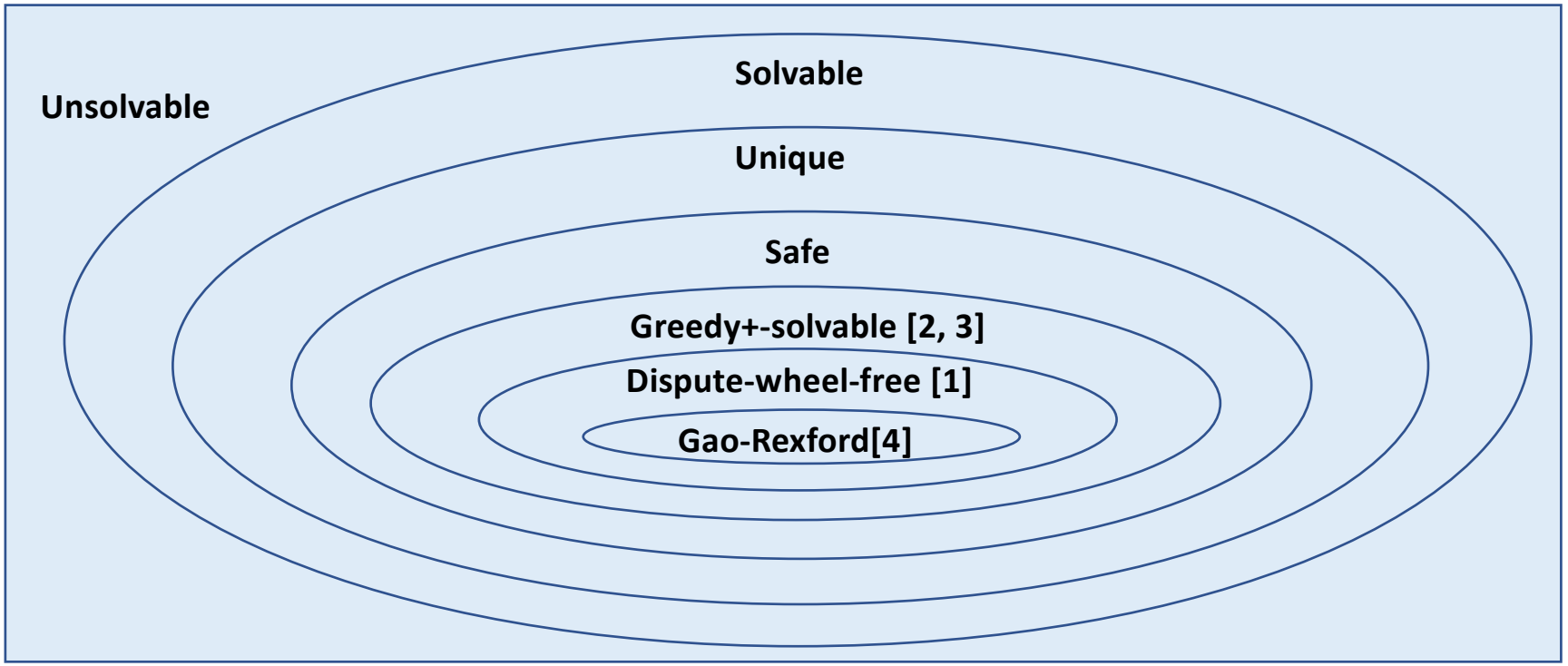
Related Work on SPP



Related Work on SPP

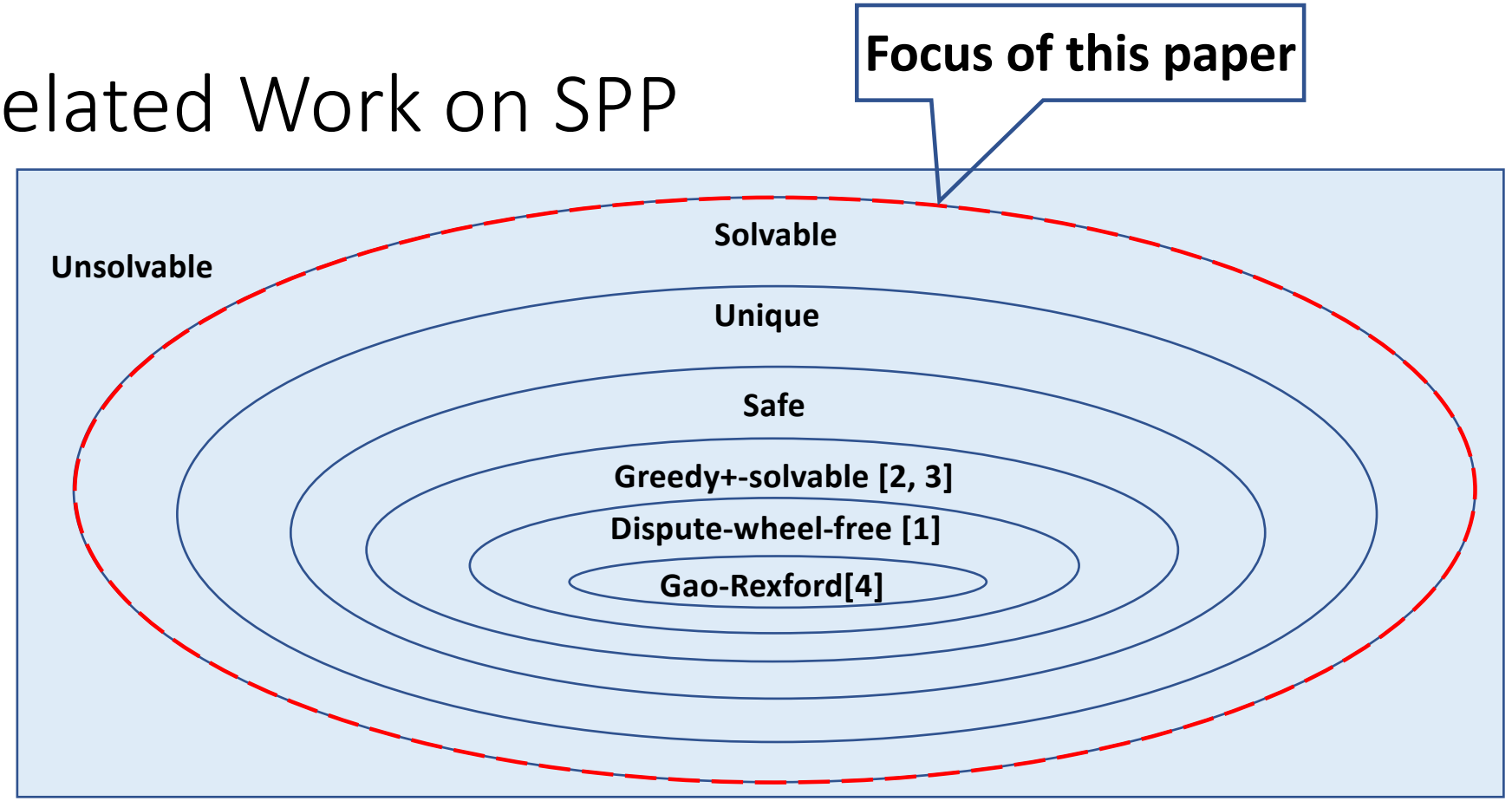


Related Work on SPP



[2] Cittadini, Luca, et al. "On the Stability of Interdomain Routing." *ACM Computing Surveys* 44, no. 4 (2012): 1-40.
[3] Cittadini, Luca, et al. "From Theory to Practice: Efficiently Checking BGP Configurations for Guaranteed Convergence." *IEEE Transactions on Network and Service Management* 8, no. 4 (2011): 387-400.
[4] Gao, Lixin et al. "Stable Internet routing without global coordination." *IEEE/ACM Transactions on networking* 9, no. 6 (2001): 681-692.

Related Work on SPP



[2] Cittadini, Luca, et al. "On the Stability of Interdomain Routing." *ACM Computing Surveys* 44, no. 4 (2012): 1-40.
[3] Cittadini, Luca, et al. "From Theory to Practice: Efficiently Checking BGP Configurations for Guaranteed Convergence." *IEEE Transactions on Network and Service Management* 8, no. 4 (2011): 387-400.
[4] Gao, Lixin et al. "Stable Internet routing without global coordination." *IEEE/ACM Transactions on networking* 9, no. 6 (2001): 681-692.

Why Focus on SPP Solvability?

- Enable new interdomain routing use cases with more flexible policies
  - E.g., collaborative interdomain routing [4], software-defined-interdomain routing [5]
- Better understand the gap between solvable SPP and safe SPP
- Help improve the efficiency of network configuration verification

[4] Pouryousef, Shahrooz, et al. "Towards Logically Centralized Interdomain Routing." In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*, pp. 739-757. 2020.
[5] Xiang, Qiao, et al. "Toward Optimal Software-Defined Interdomain Routing." In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1529-1538. IEEE, 2020.

Outline

- Background on SPP
- Our Contribution

Contribution: New SPP Solvability Results and Interdomain Use Cases

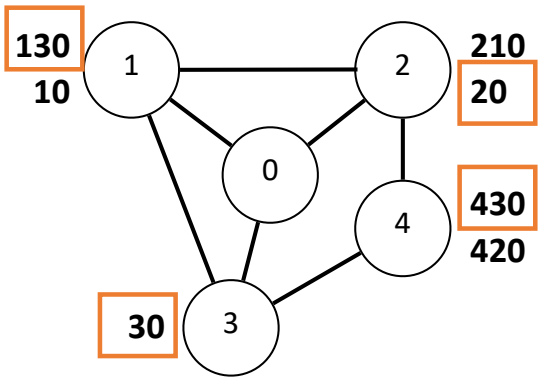
1. **Theorem**: A novel solvability digraph for reasoning SPP solvability
2. **Algorithm**: A polynomial-time heuristic solves more SPP than state-of-the-art
3. **Use case**: collaborative interdomain routing
4. **Use case**: privacy-preserving collaborative interdomain routing

Outline

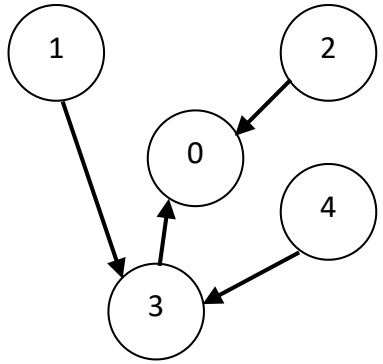
- Background on SPP
- Our Contribution
- Solvability Digraph

Properties of Stable Path Assignment

In a stable path assignment:



Good gadget

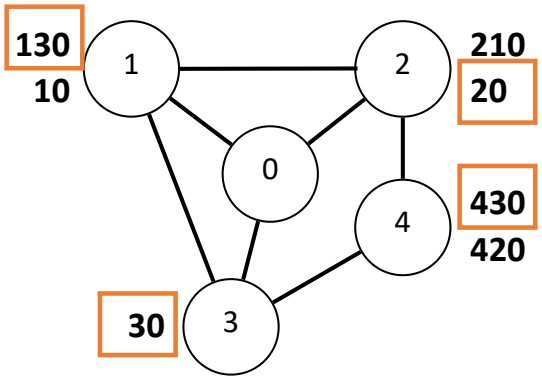


Stable path assignment for good gadget

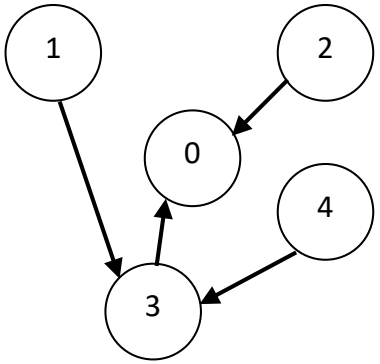
Properties of Stable Path Assignment

In a stable path assignment:

1. each node only has one path



Good gadget

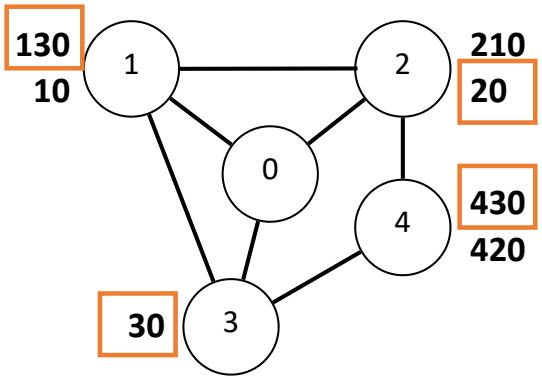


Stable path assignment for good gadget

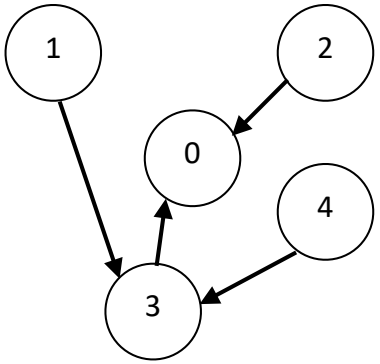
Properties of Stable Path Assignment

In a stable path assignment:

1. each node only has one path
2. each path's suffix must also be in the assignment



Good gadget

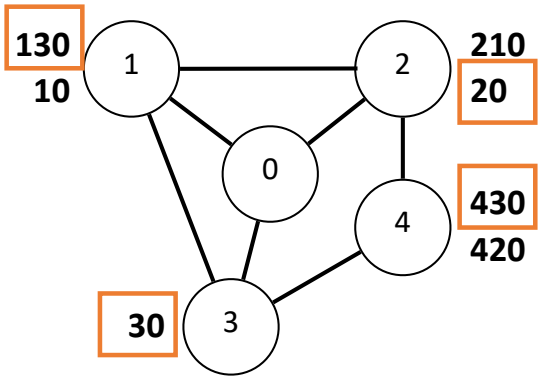


Stable path assignment for good gadget

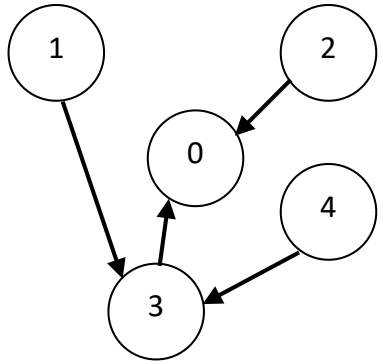
Properties of Stable Path Assignment

In a stable path assignment:

1. each node only has one path
2. each path's suffix must also be in the assignment
3. the stable path of each node is its most preferred one among the concatenation of the stable paths of all its neighbors



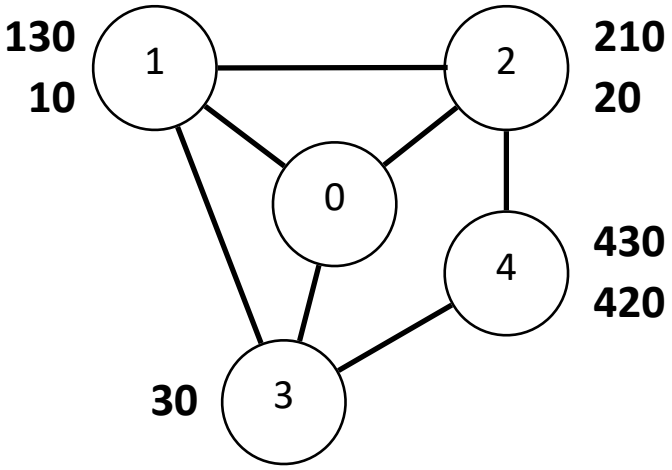
Good gadget



Stable path assignment for good gadget

Solvability Digraph

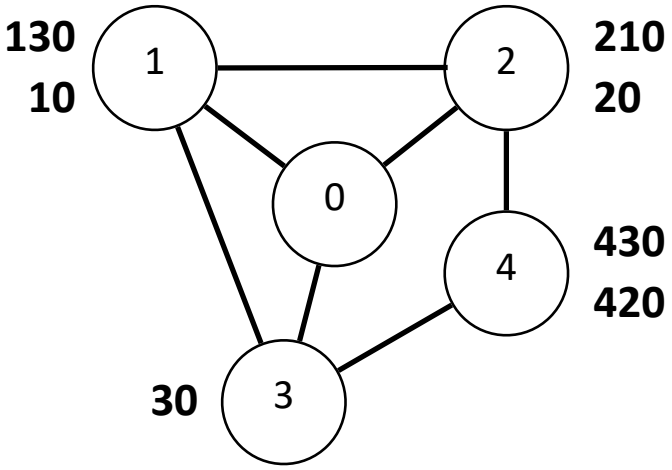
Given an SPP instance, the corresponding solvability digraph is defined as:



Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

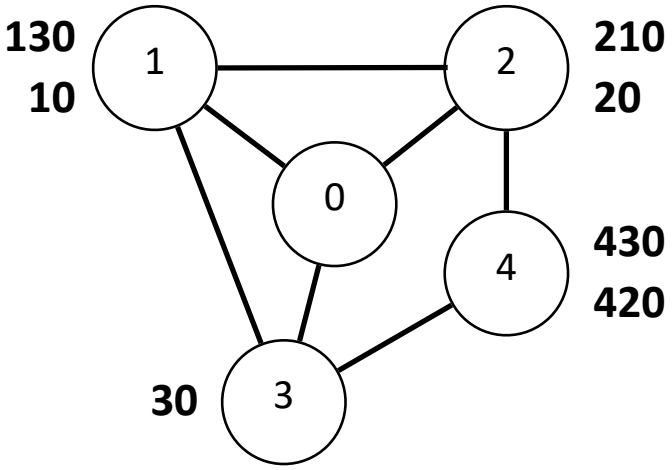
- **Vertices**: each vertex corresponds to a permitted path

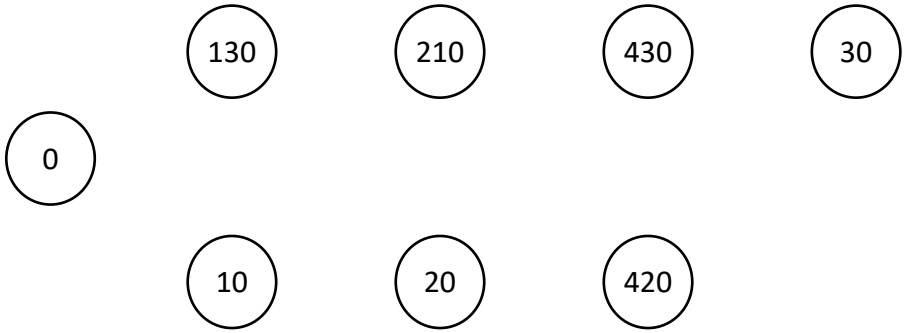


Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path

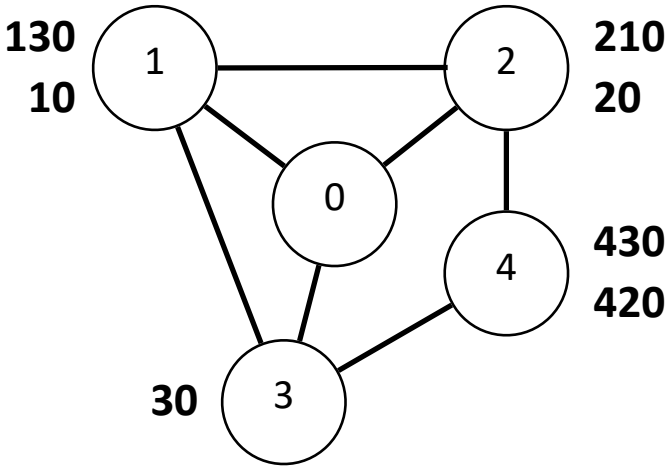


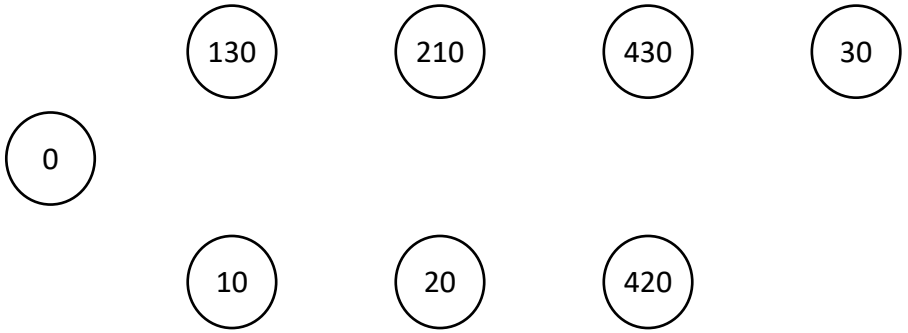


Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.



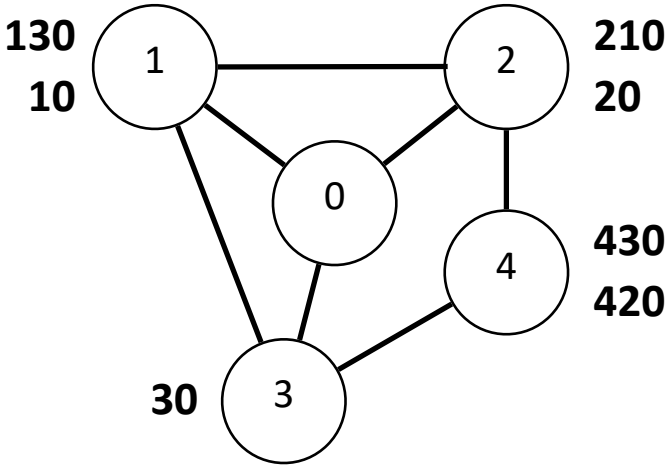


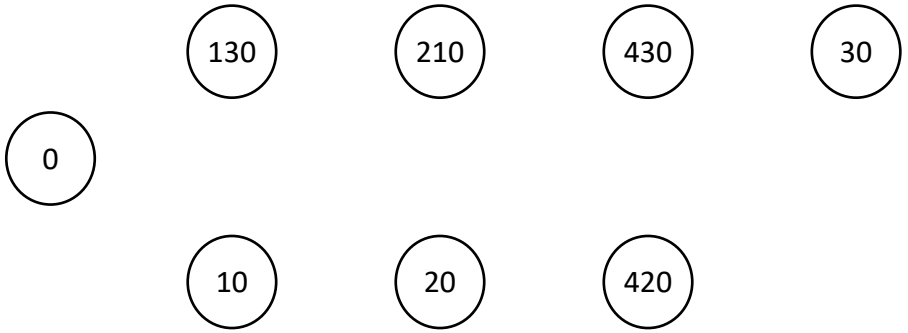
Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

Type-1 edge (p1, p2): p1 and p2 from the same node's permitted paths, and p2 is more preferred



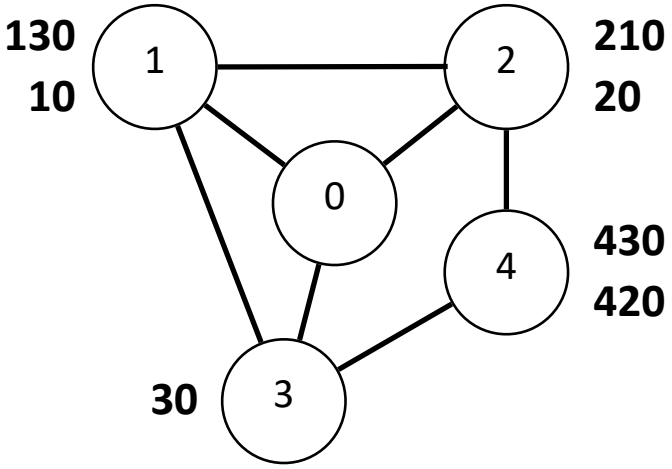


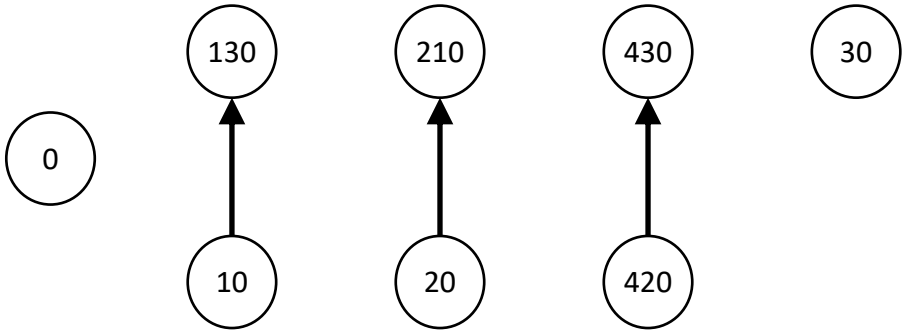
Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

Type-1 edge (p1, p2): p1 and p2 from the same node's permitted paths, and p2 is more preferred



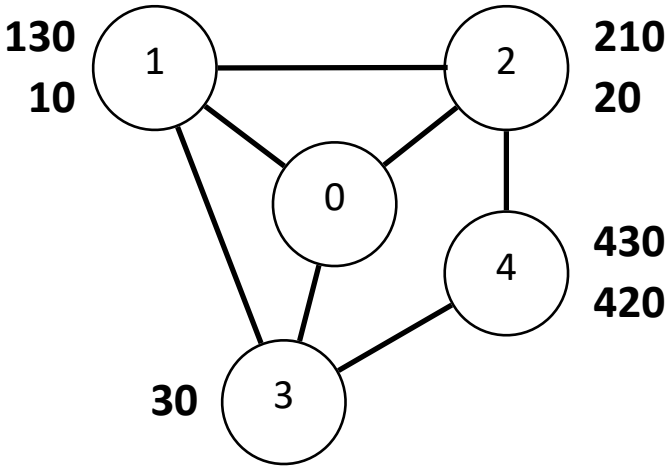


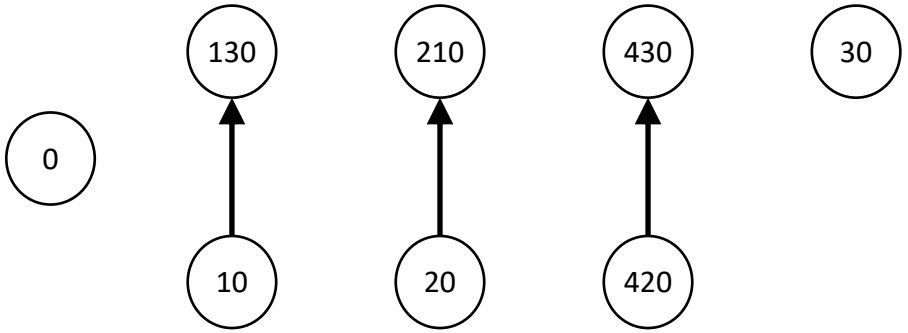
Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

Type-2 edge (p1, p2): iff there exists a p3 such that (1) p2 and p3 are from the same node, and (2) p3 is a suffix of p1



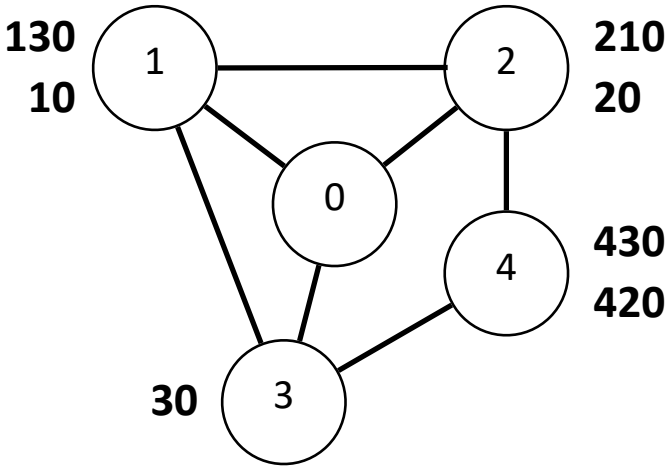


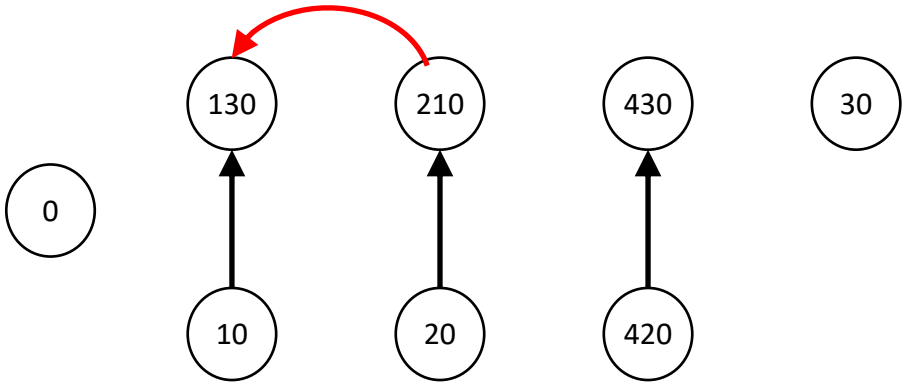
Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

Type-2 edge (p1, p2): iff there exists a p3 such that (1) p2 and p3 are from the same node, and (2) p3 is a suffix of p1



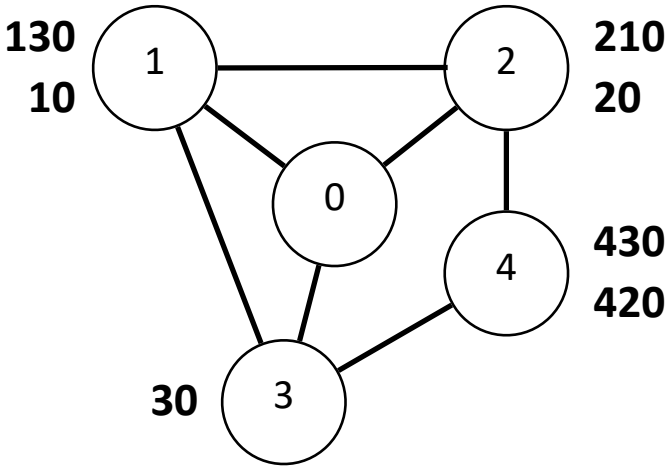


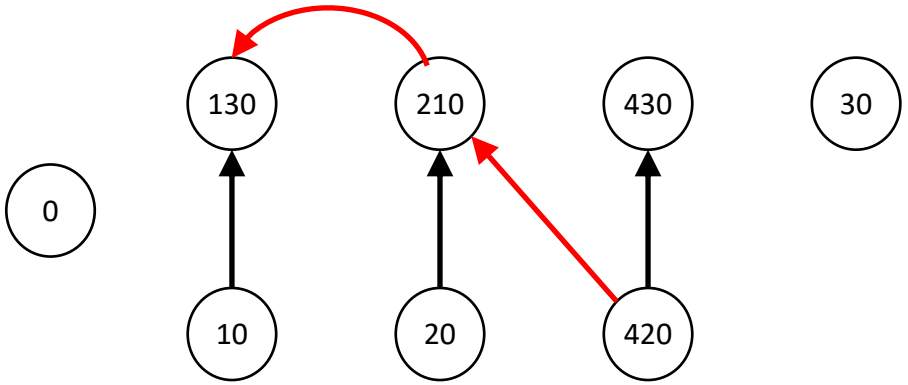
Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

Type-2 edge (p1, p2): iff there exists a p3 such that (1) p2 and p3 are from the same node, and (2) p3 is a suffix of p1



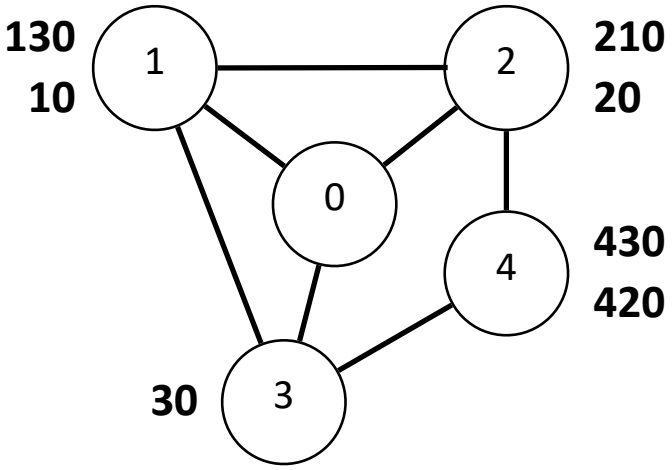


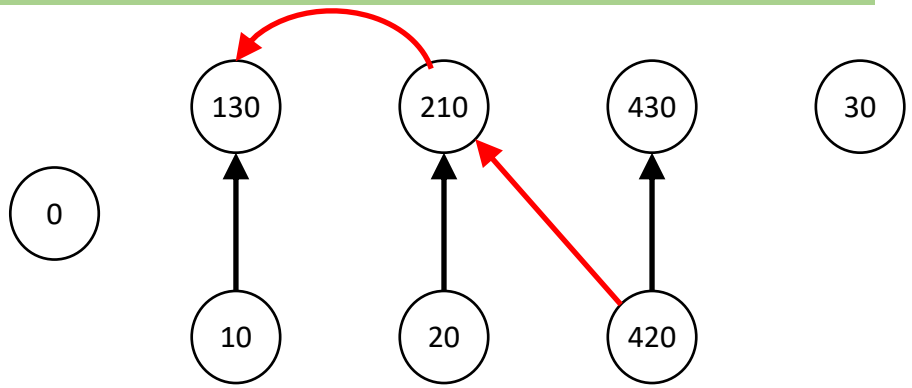
Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

Type-3 edge (p1, p2): iff there exists a p3 such that (1) p3 is from the same node as p1, but more preferred, and (2) p2 is a suffix of p3 and is 1-hop shorter.



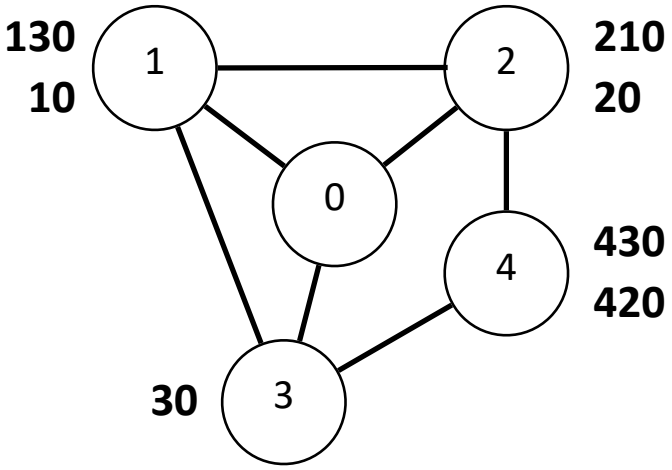


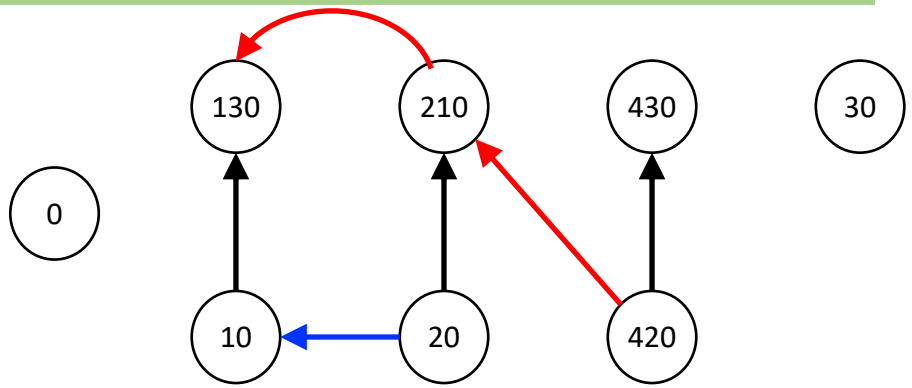
Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

Type-3 edge (p1, p2): iff there exists a p3 such that (1) p3 is from the same node as p1, but more preferred, and (2) p2 is a suffix of p3 and is 1-hop shorter.



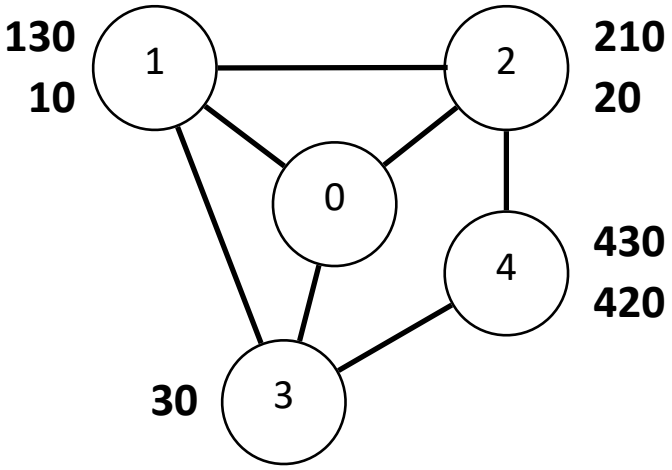


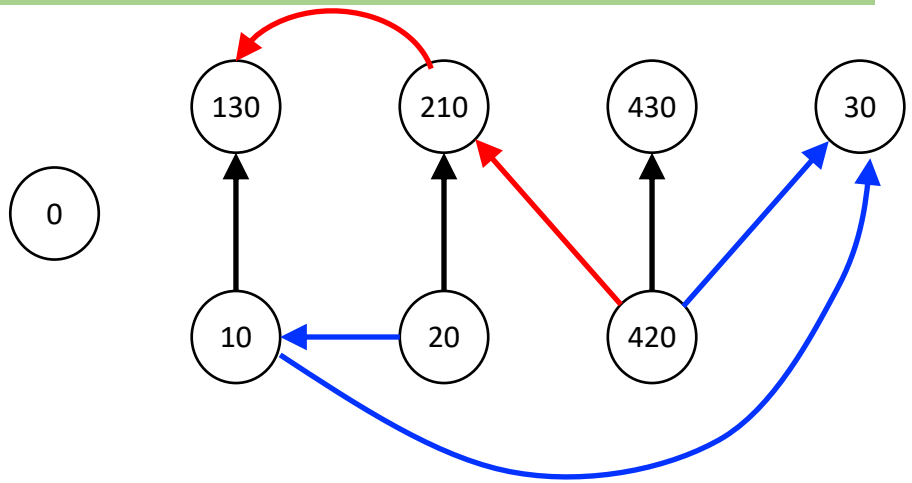
Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

Type-3 edge (p1, p2): iff there exists a p3 such that (1) p3 is from the same node as p1, but more preferred, and (2) p2 is a suffix of p3 and is 1-hop shorter.

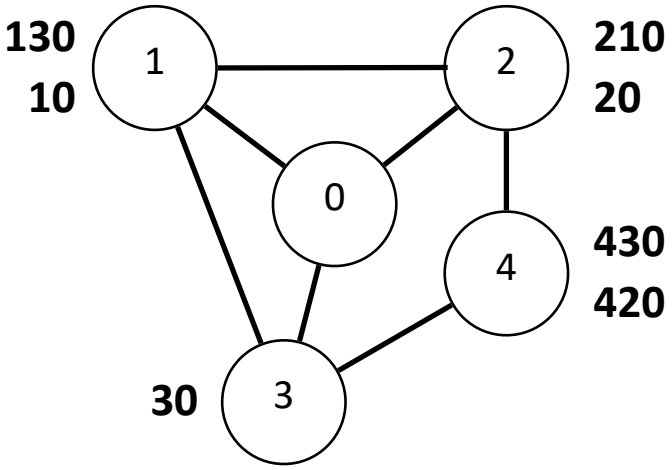


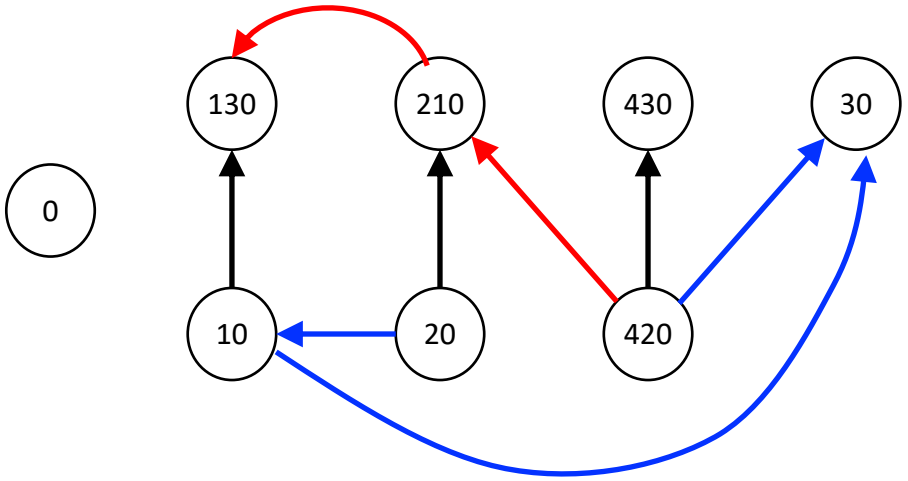


Solvability Digraph

Given an SPP instance, the corresponding solvability digraph is defined as:

- **Vertices**: each vertex corresponds to a permitted path
- **Edges**: 3 types of edges correspond to the 3 properties of SPP.

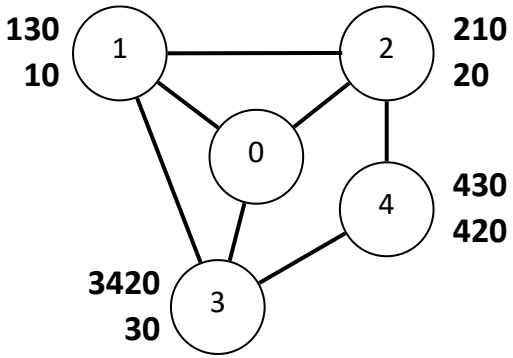




Solvability Graph - More Examples

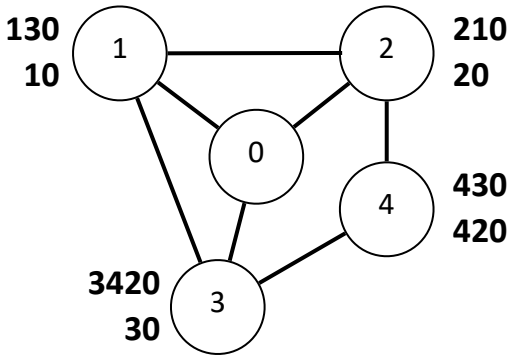
Solvability Graph - More Examples

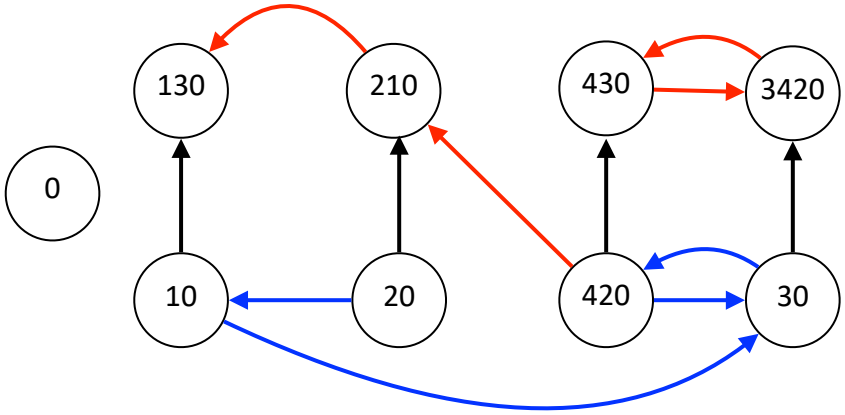
Naughty Gadget



Solvability Graph - More Examples

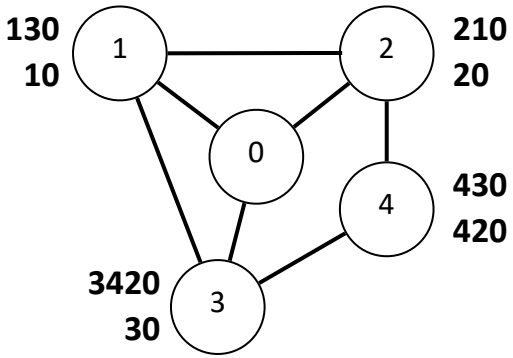
Naughty Gadget

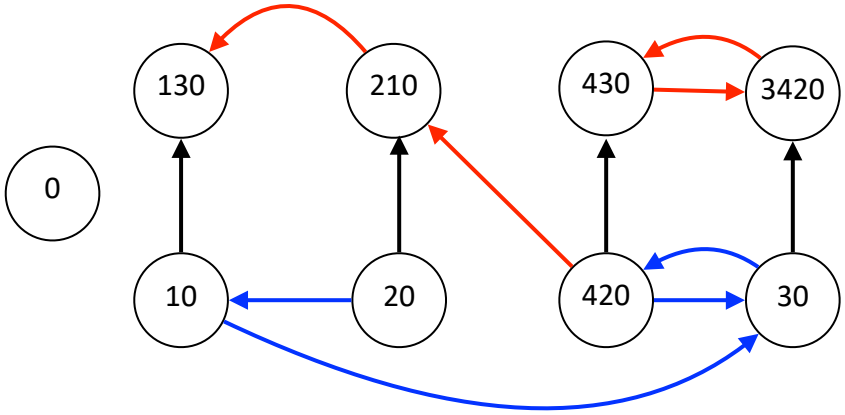




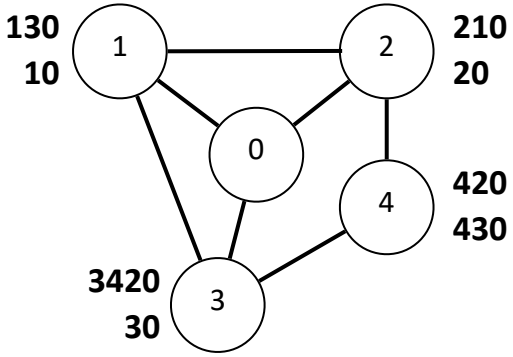
Solvability Graph - More Examples

Naughty Gadget



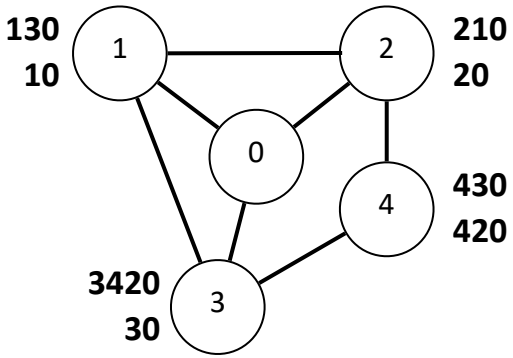


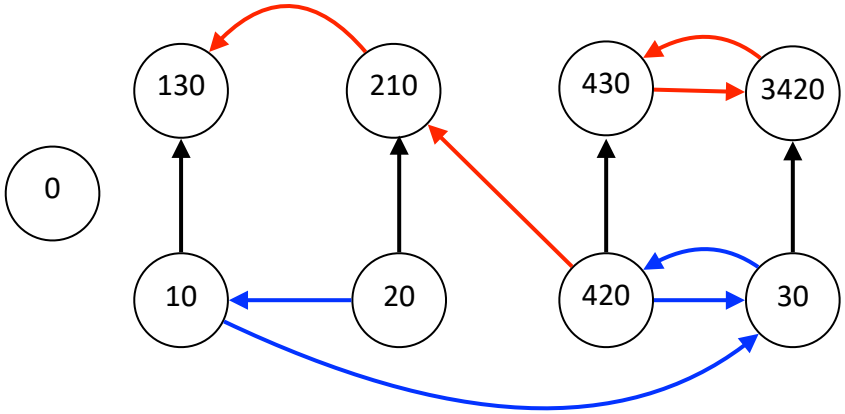
Bad Gadget



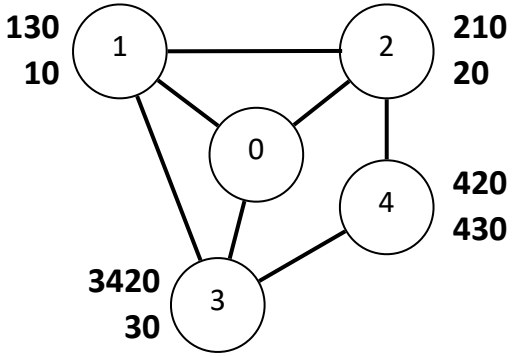
Solvability Graph - More Examples

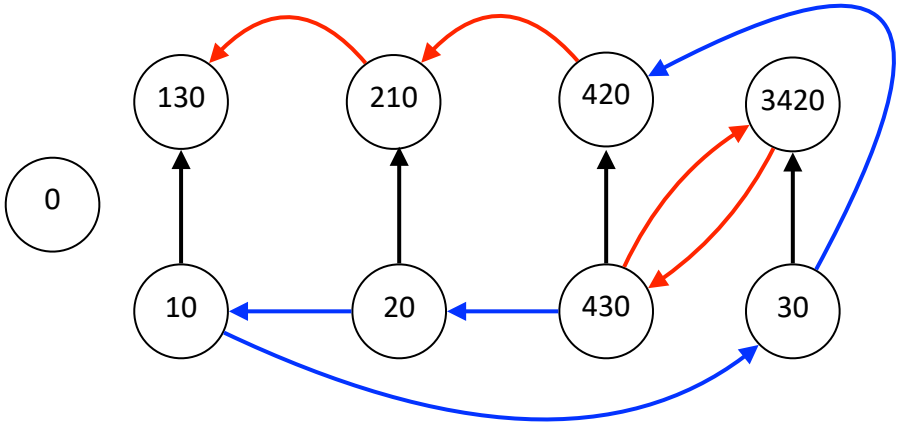
Naughty Gadget





Bad Gadget



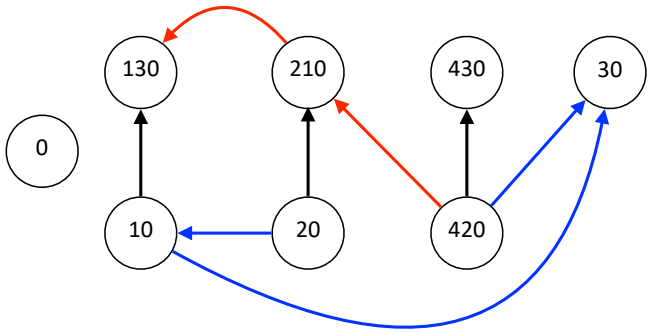


Solvability Graph and SPP Solvability

Theorem 1: Given an SPP instance with n+1 nodes, it is solvable if and only if the corresponding solvability digraph has a maximum independent set of size n+1. (proof in the paper)

Solvability Graph and SPP Solvability

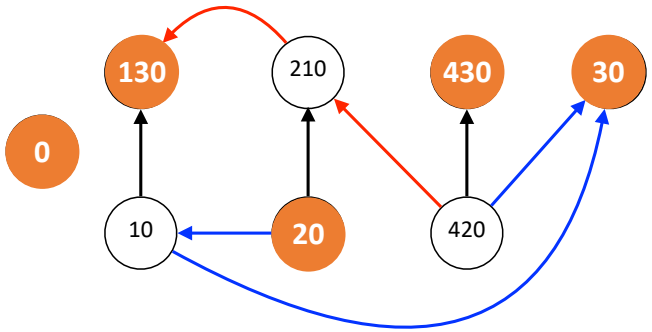
Theorem 1: Given an SPP instance with n+1 nodes, it is solvable if and only if the corresponding solvability digraph has a maximum independent set of size n+1. (proof in the paper)



Good gadget

Solvability Graph and SPP Solvability

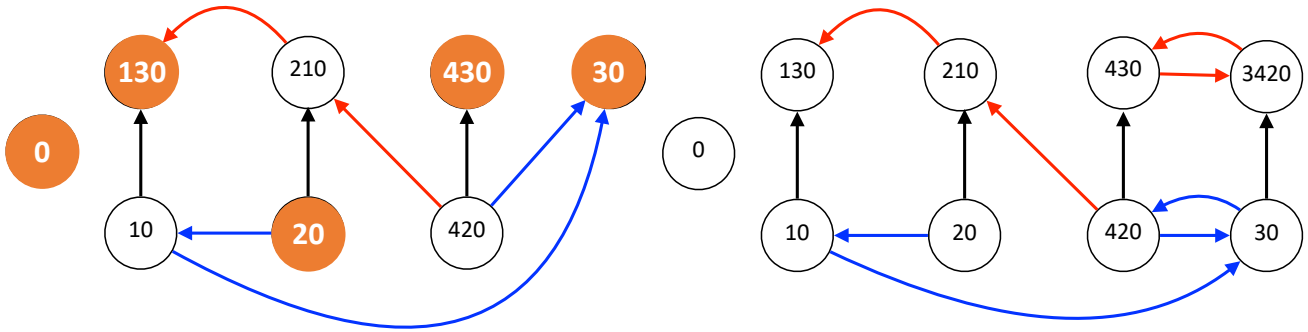
Theorem 1: Given an SPP instance with n+1 nodes, it is solvable if and only if the corresponding solvability digraph has a maximum independent set of size n+1. (proof in the paper)



Good gadget

Solvability Graph and SPP Solvability

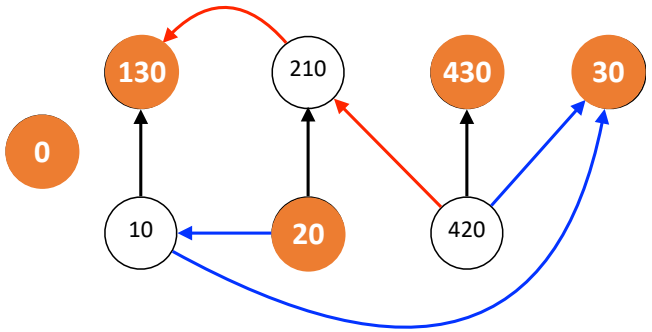
Theorem 1: Given an SPP instance with n+1 nodes, it is solvable if and only if the corresponding solvability digraph has a maximum independent set of size n+1. (proof in the paper)



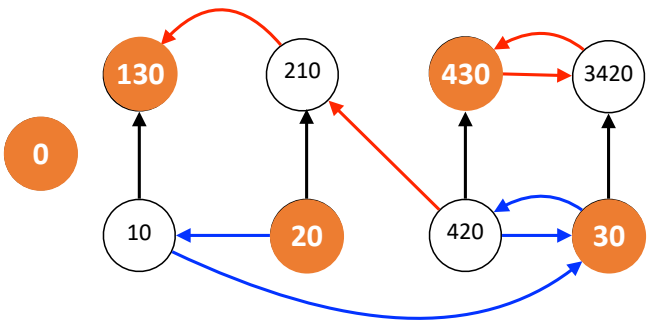
Good gadget **Naughty gadget**

Solvability Graph and SPP Solvability

Theorem 1: Given an SPP instance with n+1 nodes, it is solvable if and only if the corresponding solvability digraph has a maximum independent set of size n+1. (proof in the paper)



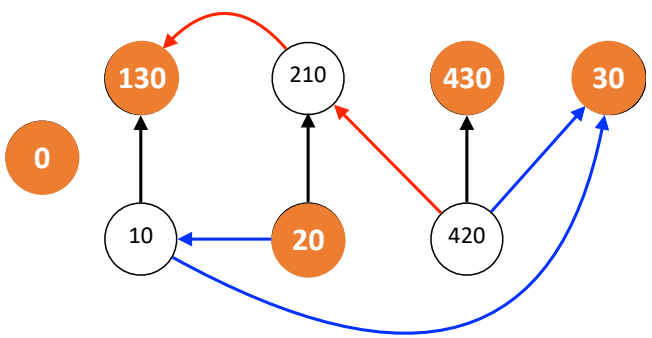
Good gadget



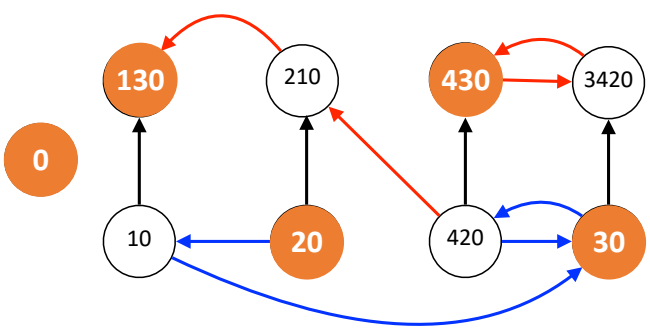
Naughty gadget

Solvability Graph and SPP Solvability

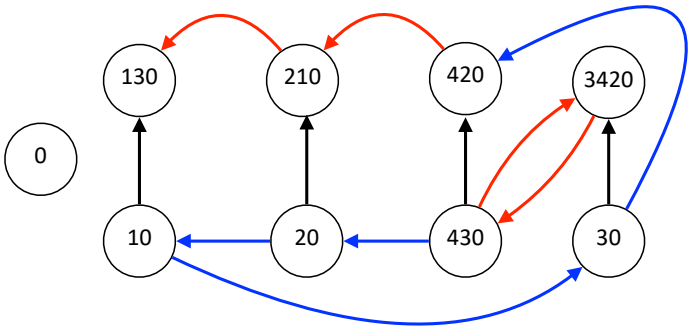
Theorem 1: Given an SPP instance with n+1 nodes, it is solvable if and only if the corresponding solvability digraph has a maximum independent set of size n+1. (proof in the paper)



Good gadget



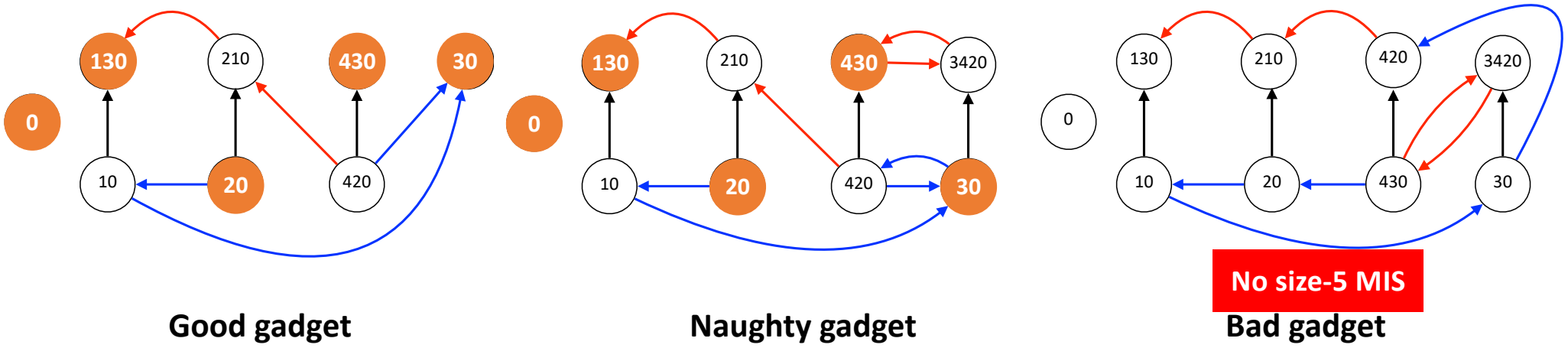
Naughty gadget



Bad gadget

Solvability Graph and SPP Solvability

Theorem 1: Given an SPP instance with n+1 nodes, it is solvable if and only if the corresponding solvability digraph has a maximum independent set of size n+1. (proof in the paper)



Good gadget **Naughty gadget** **Bad gadget**

Outline

- Background on SPP
- Our Contribution
- Solvability Digraph
- GreedyMIS: A Polynomial-Time Heuristic for SPP

GreedyMIS: Basic Idea

GreedyMIS: Basic Idea

- Iteratively add nodes into an independent set of the solvability digraph

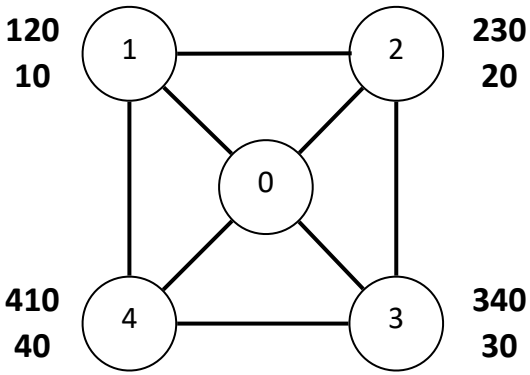
GreedyMIS: Basic Idea

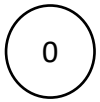
- Iteratively add nodes into an independent set of the solvability digraph
- In each iteration:
  1. add path-vertices with no out edge, if none
  2. add one path-vertex if it is the only one left for the same node in SPP, if none
  3. randomly add one path-vertex with the lowest out degree
  4. remove all added path-vertices, their neighbors and all related edges from the digraph

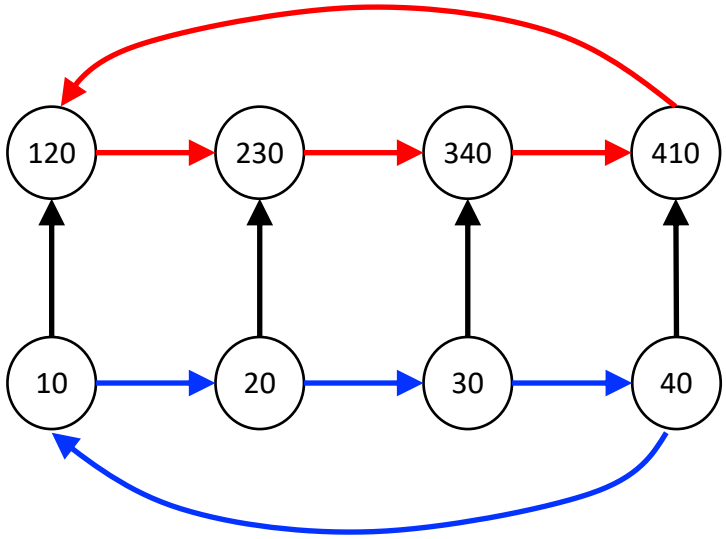
GreedyMIS: Basic Idea

- Iteratively add nodes into an independent set of the solvability digraph
- In each iteration:
  1. add path-vertices with no out edge, if none
  2. add one path-vertex if it is the only one left for the same node in SPP, if none
  3. randomly add one path-vertex with the lowest out degree
  4. remove all added path-vertices, their neighbors and all related edges from the digraph
- Terminates when
  - (**solvable**) one of size n+1 is found, or
  - (**non-solvable**) no more nodes can be added into the set

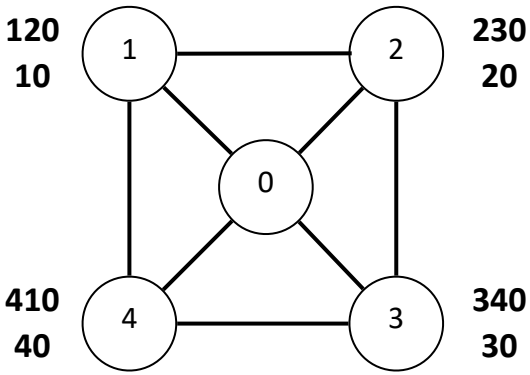
GreedyMIS: Example

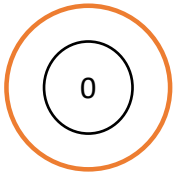


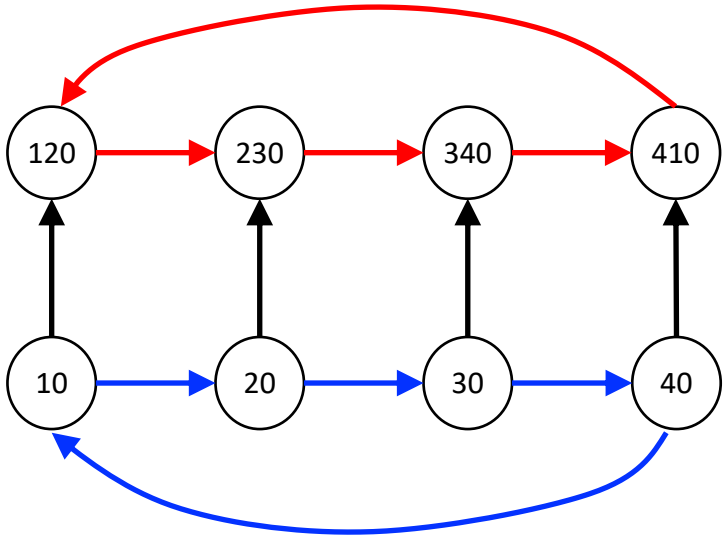




GreedyMIS: Example

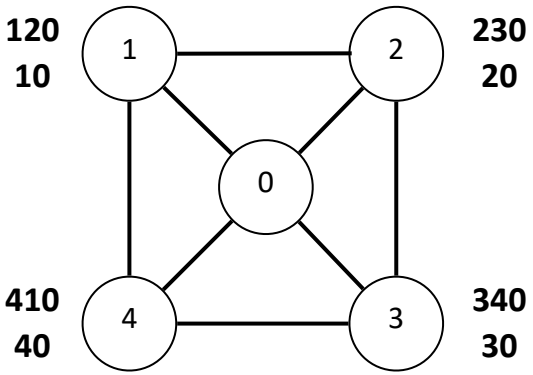


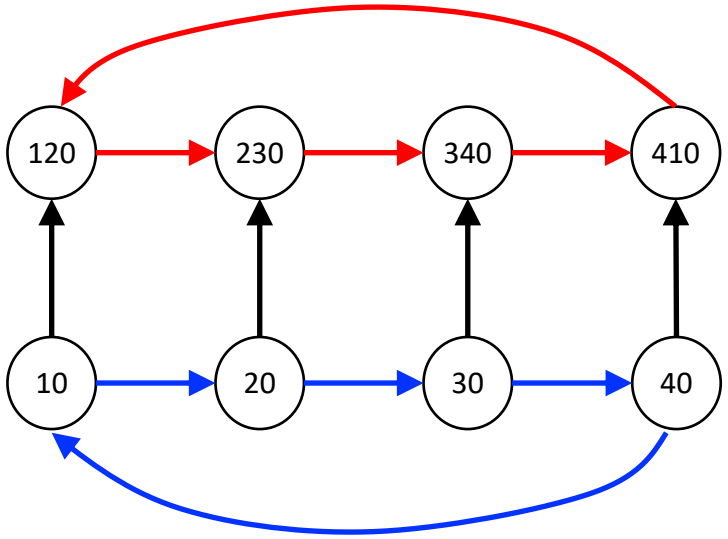




- Iteration 1: 0 is added (zero out edges)

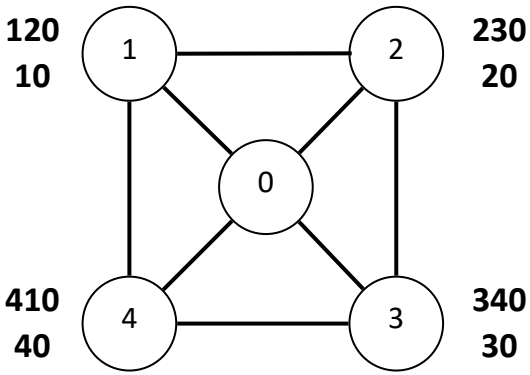
GreedyMIS: Example

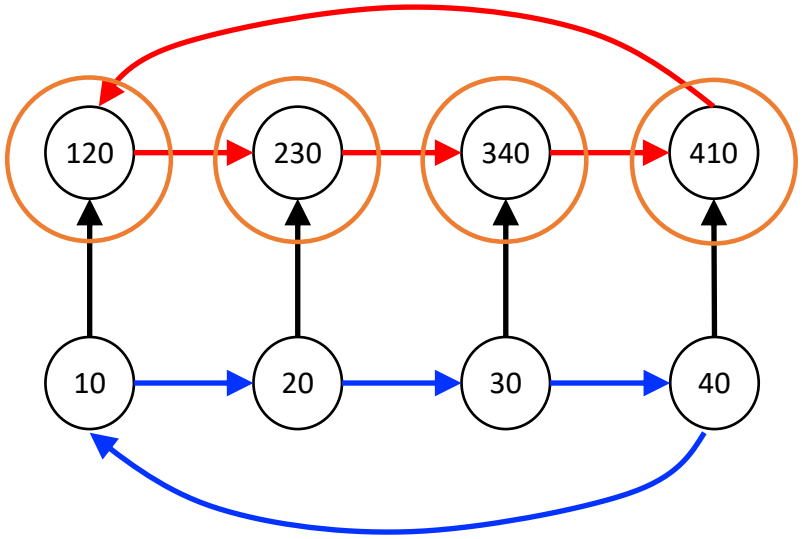




- Iteration 1: 0 is added (zero out edges)

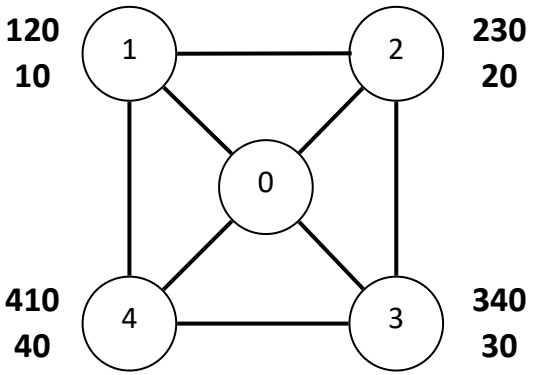
GreedyMIS: Example

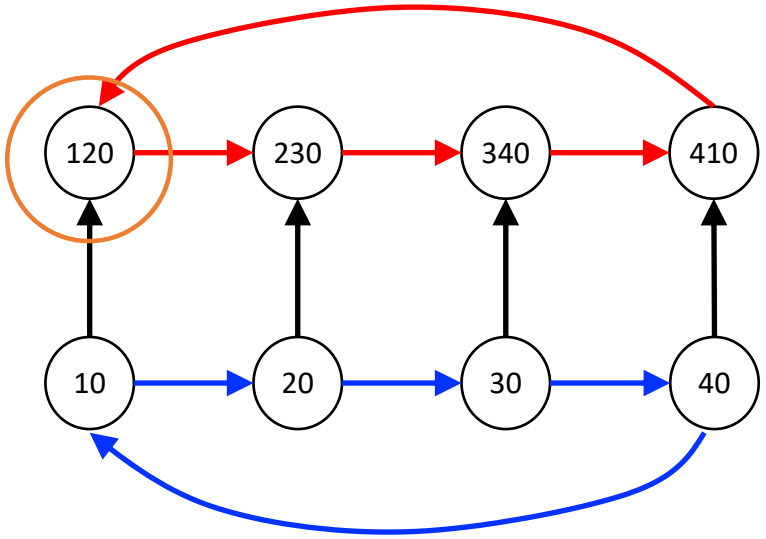




- Iteration 1: 0 is added (zero out edges)
- Iteration 2: {120, 230, 340, 410} all have out degree of 1, randomly add 120 (one with lowest out degree)

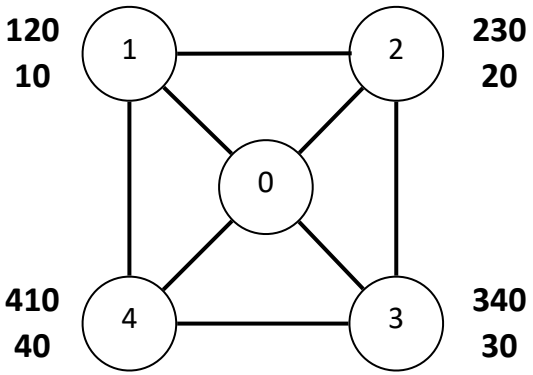
GreedyMIS: Example

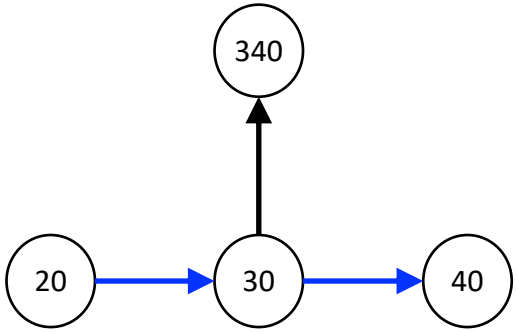




- Iteration 1: 0 is added (zero out edges)
- Iteration 2: {120, 230, 340, 410} all have out degree of 1, randomly add 120 (one with lowest out degree)

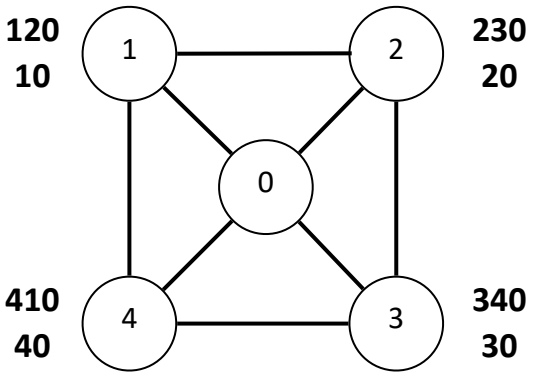
GreedyMIS: Example

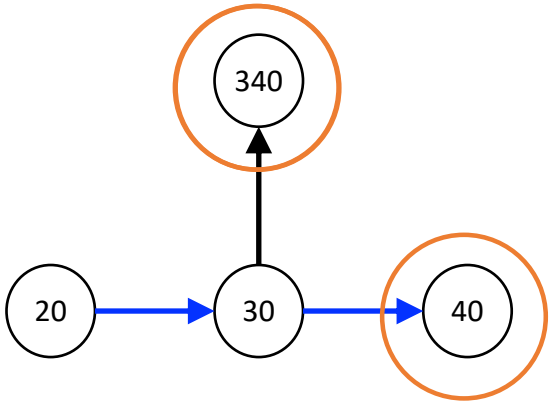




- Iteration 1: 0 is added (zero out edges)
- Iteration 2: {120, 230, 340, 410} all have out degree of 1, randomly add 120 (one with lowest out degree)

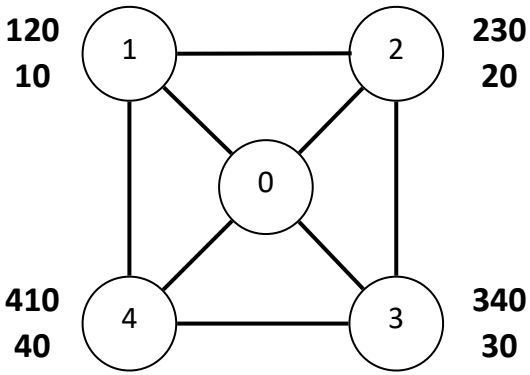
GreedyMIS: Example

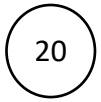




- Iteration 1: 0 is added (zero out edges)
- Iteration 2: {120, 230, 340, 410} all have out degree of 1, randomly add 120 (one with lowest out degree)
- Iteration 3: 340, 40 are added (zero out edges)

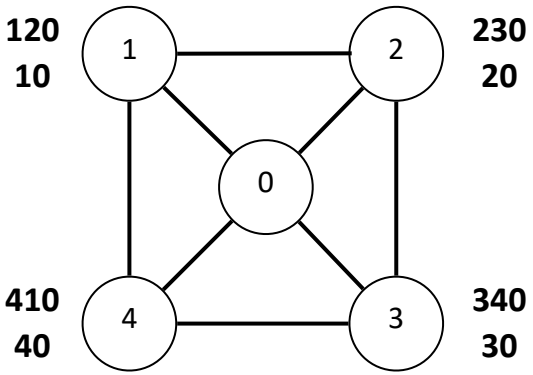
GreedyMIS: Example

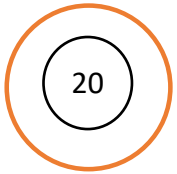




- Iteration 1: 0 is added (zero out edges)
- Iteration 2: {120, 230, 340, 410} all have out degree of 1, randomly add 120 (one with lowest out degree)
- Iteration 3: 340, 40 are added (zero out edges)

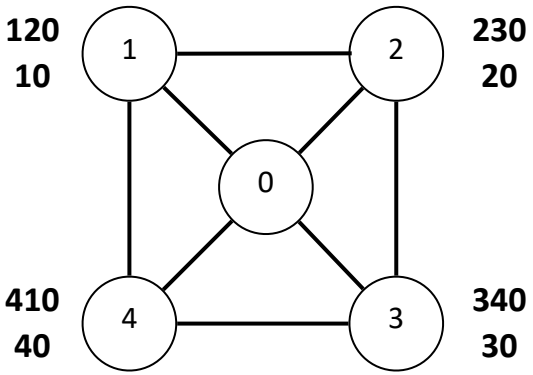
GreedyMIS: Example





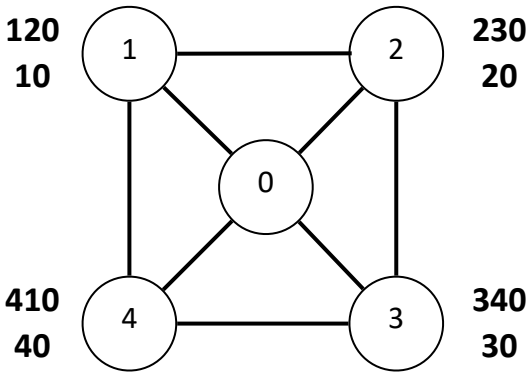
- Iteration 1: 0 is added (zero out edges)
- Iteration 2: {120, 230, 340, 410} all have out degree of 1, randomly add 120 (one with lowest out degree)
- Iteration 3: 340, 40 are added (zero out edges)
- Iteration 4: 20 is added (zero out edges)

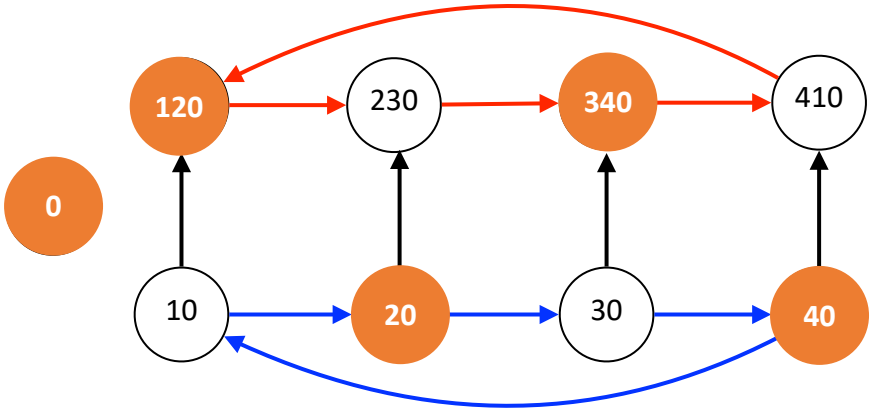
GreedyMIS: Example



- Iteration 1: 0 is added (zero out edges)
- Iteration 2: {120, 230, 340, 410} all have out degree of 1, randomly add 120 (one with lowest out degree)
- Iteration 3: 340, 40 are added (zero out edges)
- Iteration 4: 20 is added (zero out edges)

GreedyMIS: Example





- Iteration 1: 0 is added (zero out edges)
- Iteration 2: {120, 230, 340, 410} all have out degree of 1, randomly add 120 (one with lowest out degree)
- Iteration 3: 340, 40 are added (zero out edges)
- Iteration 4: 20 is added (zero out edges)
- **Final result**: {0, 120, 340, 40, 20}

GreedyMIS: Analysis

Theorem 2: Any SPP instance that is solvable by Greedy+ [3], , the best known polynomial-time heuristic algorithm for SPP, is also solvable by GreedyMIS. (proof in the paper)

GreedyMIS: Analysis

Theorem 2: Any SPP instance that is solvable by Greedy+ [3], , the best known polynomial-time heuristic algorithm for SPP, is also solvable by GreedyMIS. (proof in the paper)

Theorem 3: GreedyMIS solves strictly more SPP instances than Greedy+.

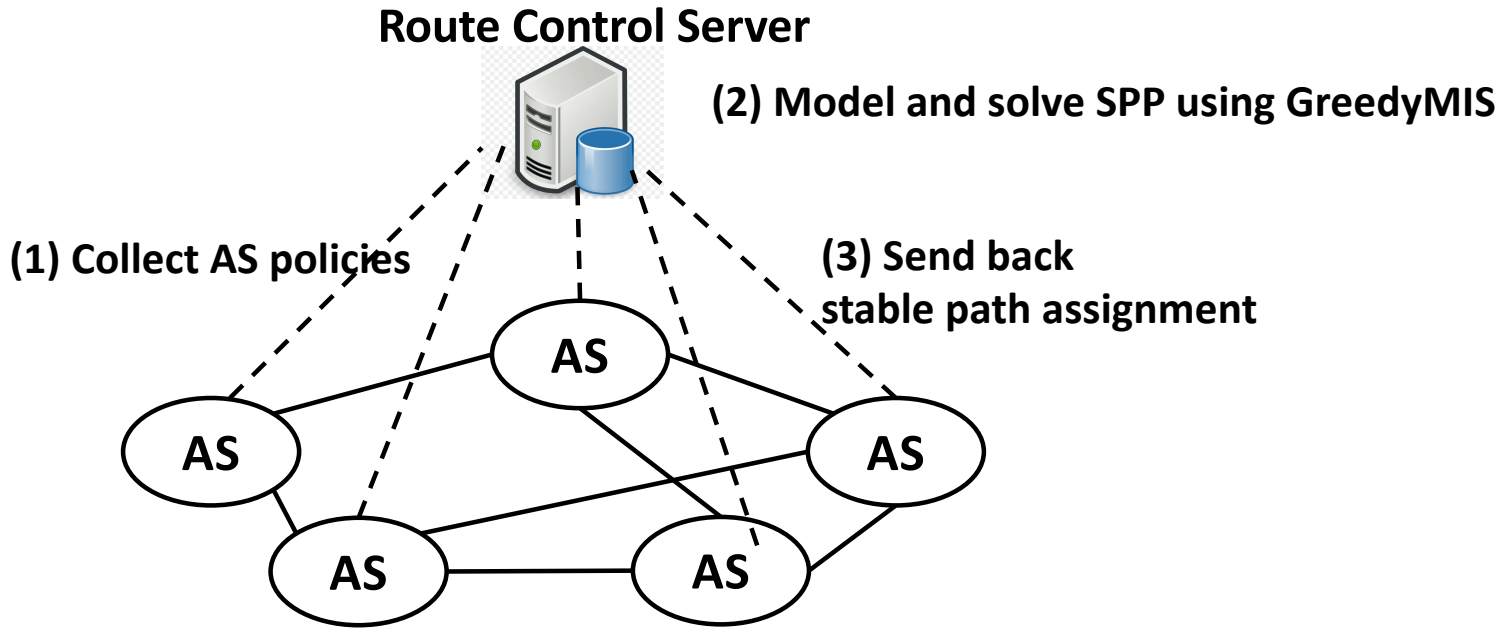
- A direct result from Theorem 2 and the example in the last slide.

Outline

- Background on SPP
- Our Contribution
- Solvability Digraph
- GreedyMIS: A Polynomial-Time Heuristic for SPP
- Use Cases

Use Case 1: Collaborative Interdomain Routing

- **Advantage**: allow more flexible routing policies, i.e., doesn't haven't to be a "safe" SPP
- **Scenario**: collaborative science networks such as Large Hadron Collider



Use Case 2: Privacy-Preserving, Collaborative Interdomain Routing

- **Scenario**: multi-domain networks desire more flexible policies, but still want to keep policies private

Use Case 2: Privacy-Preserving, Collaborative Interdomain Routing

- **Scenario**: multi-domain networks desire more flexible policies, but still want to keep policies private
- **Design**: each AS exposes its export policies since it can be inferred, but keeps selection policy private

Use Case 2: Privacy-Preserving, Collaborative Interdomain Routing

- **Scenario**: multi-domain networks desire more flexible policies, but still want to keep policies private
- **Design**: each AS exposes its export policies since it can be inferred, but keeps selection policy private
- **Protocol**: an privacy-preserving version of GreedyMIS that employs linear secret sharing

Outline

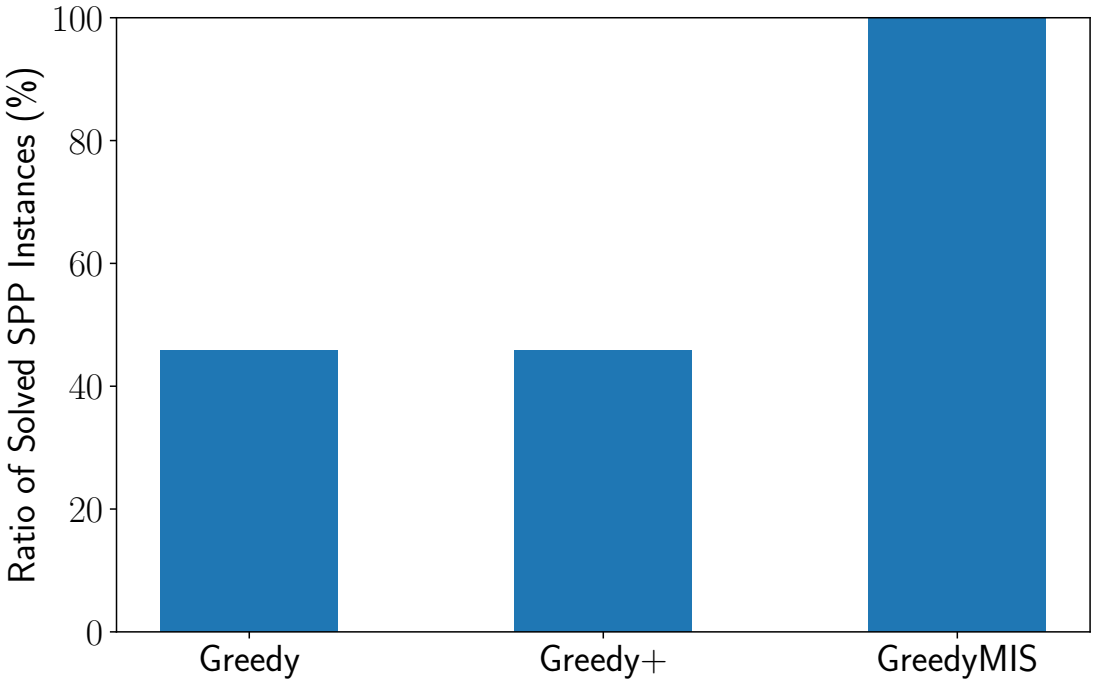
- Background on SPP
- Our Contribution
- Solvability Digraph
- GreedyMIS: A Polynomial-Time Heuristic for SPP
- Use Cases
- Performance Evaluation

Performance Evaluation: Settings

- **Topology**: real AS-level topologies from CAIDA Internet topology dataset
- **AS export policies**: (1) C/P relationship, (2) blacklist ASes, (3) forbidden segments.
- **SPP instances**: 11-50 ASes, 8-12k permitted paths
- **Comparison**: GreedyMIS vs. Greedy [1] and Greedy+ [3]

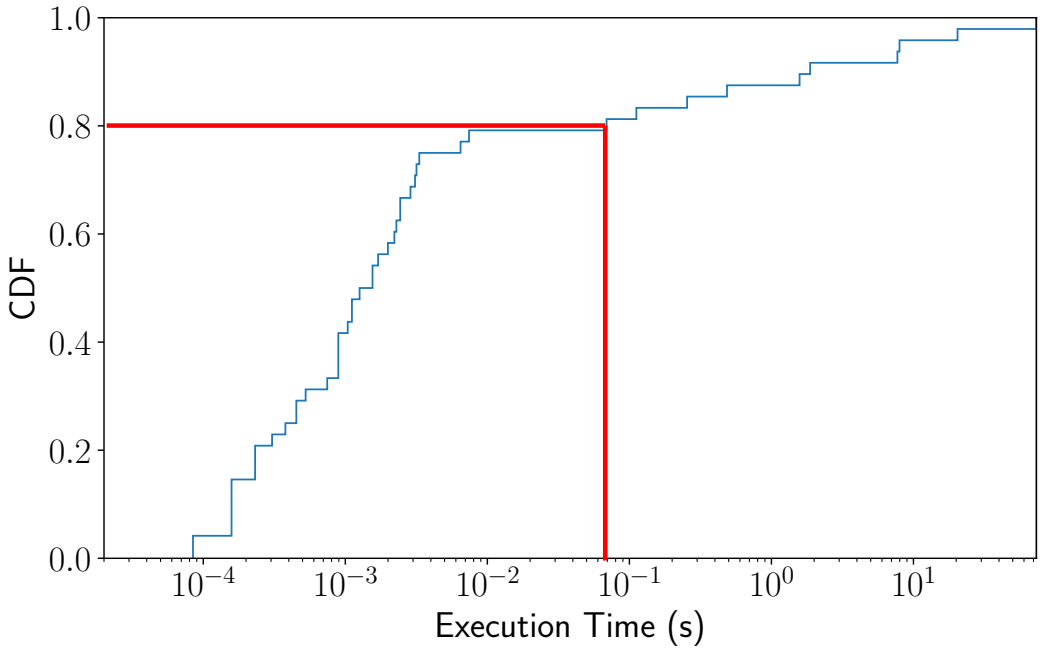
Results: Solvability of GreedyMIS

- GreedyMIS solves all SPP instances in the experiment, while the other two algorithms solves less than 50%.



Results: Efficiency of GreedyMIS

- In 80% of the experiments, GreedyMIS solves SPP in no more than 0.01s



Conclusion and Future Work

- Propose the solvability digraph data structure to reason about SPP solvability
- Design GreedyMIS, a polynomial-time heuristic that solves strictly more SPP instances than known state-of-the-art
- Tailor GreedyMIS to important interdomain routing use cases
- Demonstrate the efficiency and potentials of GreedyMIS via evaluation on real topologies

Conclusion and Future Work

- Propose the solvability digraph data structure to reason about SPP solvability
- Design GreedyMIS, a polynomial-time heuristic that solves strictly more SPP instances than known state-of-the-art
- Tailor GreedyMIS to important interdomain routing use cases
- Demonstrate the efficiency and potentials of GreedyMIS via evaluation on real topologies

Future Work

- Generic exact/approximated MIS algorithm for solving SPP
- Apply the SPP-MIS theorem to network configuration verification

Backup Slides