Temporal Reachability Minimization: Delaying vs. Deleting

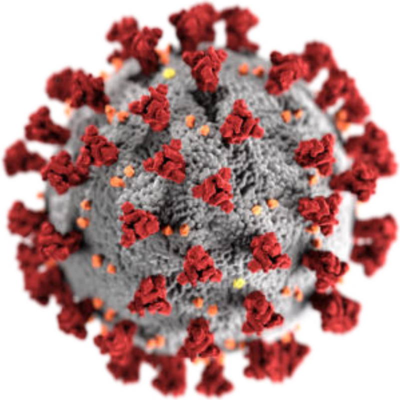
Hendrik Molter[1] Malte Renken[2] Philipp Zschoche[2]

[1] Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Israel
[2] Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Germany

Algorithmic Aspects of Temporal Graphs IV

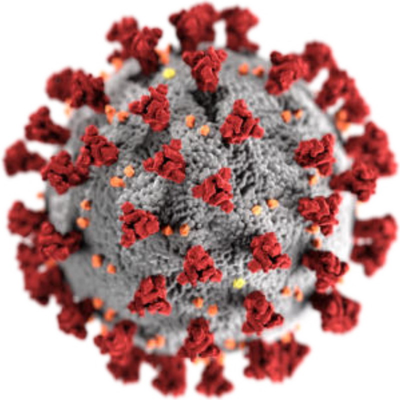
Extended abstract to appear in proceedings of MFCS 2021

Motivation: Disease Spreading



Scenario: Diseases spread through physical contact.

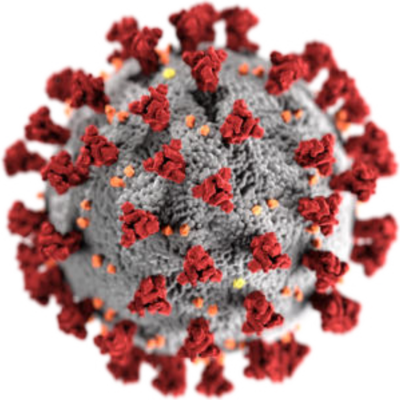
Motivation: Disease Spreading



Scenario: Diseases spread through physical contact.

Goal: Minimize potential infections to contain disease:

Motivation: Disease Spreading

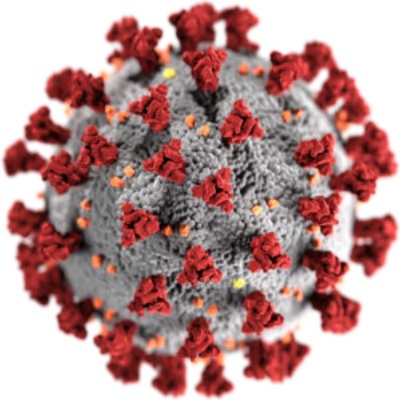


Scenario: Diseases spread through physical contact.

Goal: Minimize potential infections to contain disease:

- By **removing** interactions.

Motivation: Disease Spreading

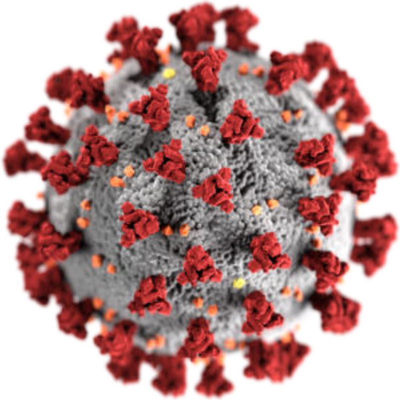


Scenario: Diseases spread through physical contact.

Goal: Minimize potential infections to contain disease:

- By **removing** interactions.
- By **delaying** interactions.

Motivation: Disease Spreading

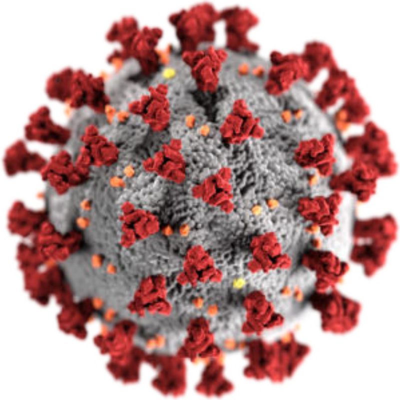


Scenario: Diseases spread through physical contact.

Goal: Minimize potential infections to contain disease:

- By **removing** interactions.
- By **delaying** interactions.
- By **reordering** interaction patterns.

Motivation: Disease Spreading



Scenario: Diseases spread through physical contact.

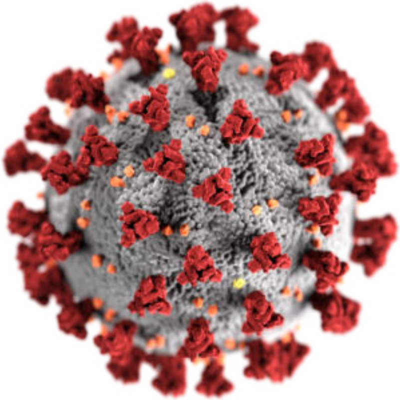
Goal: Minimize potential infections to contain disease:

- By **removing** interactions.
- By **delaying** interactions.
- By **reordering** interaction patterns.

Assumptions:

- Known outbreak.

Motivation: Disease Spreading



Scenario: Diseases spread through physical contact.

Goal: Minimize potential infections to contain disease:

- By **removing** interactions.
- By **delaying** interactions.
- By **reordering** interaction patterns.

Assumptions:

- Known outbreak.
- Unknown outbreak.

Related Work

Minimizing Reachability by Time-Edge Modification

Modification	Known Outbreak	Unknown Outbreak

Related Work

Minimizing Reachability by Time-Edge Modification

Modification	Known Outbreak	Unknown Outbreak
deleting	Our Work [MFCS ’21]	Enright & Meeks [Algorithmica ’18], Enright, Meeks, Mertzios, Zamaraev [JCSS ’21]

Related Work

Minimizing Reachability by Time-Edge Modification

Modification	Known Outbreak	Unknown Outbreak
deleting	Our Work [MFCS ’21]	Enright & Meeks [Algorithmica ’18], Enright, Meeks, Mertzios, Zamaraev [JCSS ’21]
delaying	Deligkas & Potapov [AAAI ’20], Our Work [MFCS ’21]	?

Related Work

Minimizing Reachability by Time-Edge Modification

Modification	Known Outbreak	Unknown Outbreak
deleting	Our Work [MFCS ’21]	Enright & Meeks [Algorithmica ’18], Enright, Meeks, Mertzios, Zamaraev [JCSS ’21]
delaying	Deligkas & Potapov [AAAI ’20], Our Work [MFCS ’21]	?
reordering	?	Enright, Meeks, Skerman [JCSS ’21]

Related Work

Minimizing Reachability by Time-Edge Modification

Modification	Known Outbreak	Unknown Outbreak
deleting	Our Work [MFCS '21]	Enright & Meeks [Algorithmica '18], Enright, Meeks, Mertzios, Zamaraev [JCSS '21]
delaying	Deligkas & Potapov [AAAI '20], Our Work [MFCS '21]	?
reordering	?	Enright, Meeks, Skerman [JCSS '21]
merging	Deligkas & Potapov [AAAI '20]	?

Temporal Graphs and Temporal Reachability: Definition

Temporal Graph

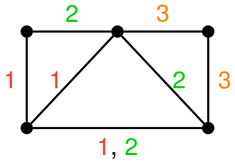
A **temporal graph** $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.

Temporal Graphs and Temporal Reachability: Definition

Temporal Graph

A **temporal graph** $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.

$\mathcal{G}$:

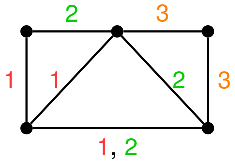


Temporal Graphs and Temporal Reachability: Definition

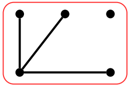
Temporal Graph

A **temporal graph** $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.

$\mathcal{G}$:



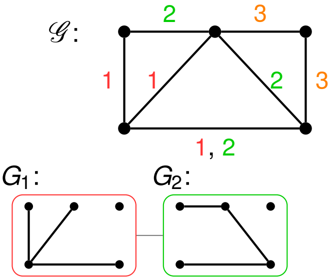
G_1:



Temporal Graphs and Temporal Reachability: Definition

Temporal Graph

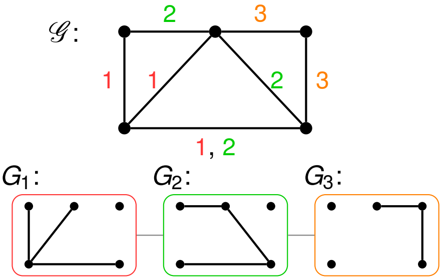
A **temporal graph** $\mathcal{G} = (V, (E_i)_{i\in[\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.



Temporal Graphs and Temporal Reachability: Definition

Temporal Graph

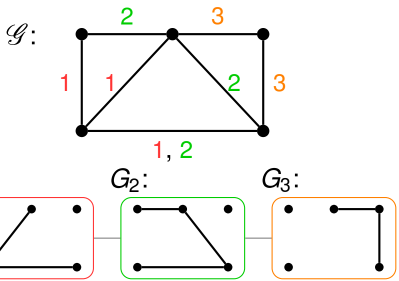
A **temporal graph** $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.



Temporal Graphs and Temporal Reachability: Definition

Temporal Graph

A **temporal graph** $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.



Temporal (s, z)-Path

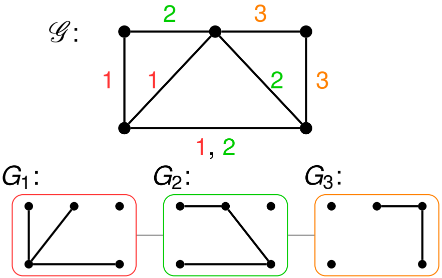
Sequence of time edges forming a path from s to z that have:

- increasing time stamps (strict).
- non-decreasing time stamps (non-strict).

Temporal Graphs and Temporal Reachability: Definition

Temporal Graph

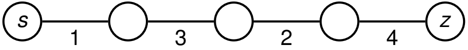
A **temporal graph** $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.



Temporal (s, z)-Path

Sequence of time edges forming a path from s to z that have:

- increasing time stamps (strict).
- non-decreasing time stamps (non-strict).

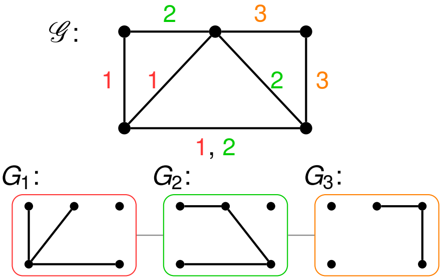


Not a temporal path.

Temporal Graphs and Temporal Reachability: Definition

Temporal Graph

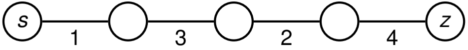
A **temporal graph** $\mathcal{G} = (V, (E_i)_{i\in[\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.



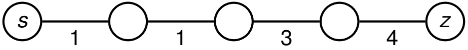
Temporal (s, z)-Path

Sequence of time edges forming a path from s to z that have:

- increasing time stamps (strict).
- non-decreasing time stamps (non-strict).



Not a temporal path.

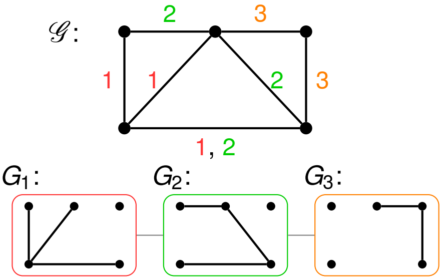


Non-strict temporal path. (Not strict.)

Temporal Graphs and Temporal Reachability: Definition

Temporal Graph

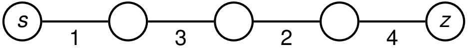
A **temporal graph** $\mathcal{G} = (V, (E_i)_{i \in [\ell]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\ell$ over V, where ℓ is the lifetime of $\mathcal{G}$.



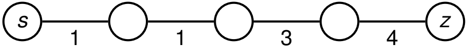
Temporal (s, z)-Path

Sequence of time edges forming a path from s to z that have:

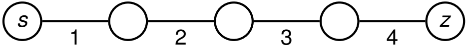
- increasing time stamps (strict).
- non-decreasing time stamps (non-strict).



Not a temporal path.



Non-strict temporal path. (Not strict.)



Temporal path (both strict and non-strict).

Minimizing Reachability by Deleting

Minimizing Reachability by Deleting

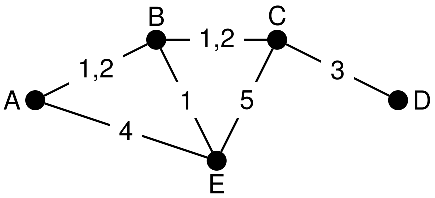
Given some initial outbreak S, can we **delete** up to k time-edges such that at most r vertices eventually contract the disease?

Minimizing Reachability by Deleting

Minimizing Reachability by Deleting

Given some initial outbreak S, can we **delete** up to k time-edges such that at most r vertices eventually contract the disease?

Example: $S = \{A\}$, $k = 2$, $r = 2$.

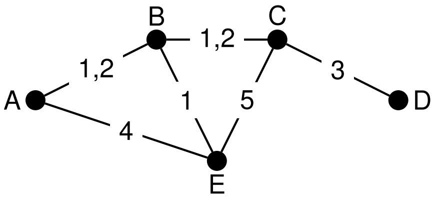


Minimizing Reachability by Deleting

Minimizing Reachability by Deleting

Given some initial outbreak S, can we **delete** up to k time-edges such that at most r vertices eventually contract the disease?

Example: $S = \{A\}$, $k = 2$, $r = 2$.



Results:

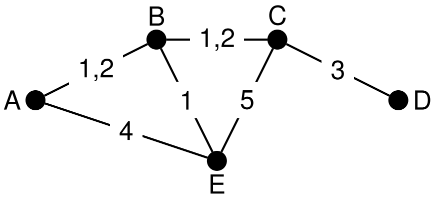
- Polytime solvable on trees.

Minimizing Reachability by Deleting

Minimizing Reachability by Deleting

Given some initial outbreak S, can we **delete** up to k time-edges such that at most r vertices eventually contract the disease?

Example: $S = \{A\}$, $k = 2$, $r = 2$.



Results:

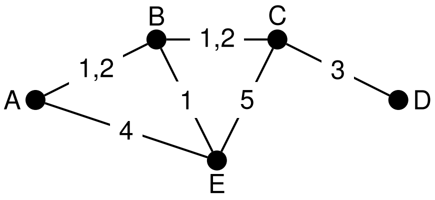
- Polytime solvable on trees.
- NP-hard and W[1]-hard wrt. k for lifetime two. Deligkas & Potapov [AAAI '20]

Minimizing Reachability by Deleting

Minimizing Reachability by Deleting

Given some initial outbreak S, can we **delete** up to k time-edges such that at most r vertices eventually contract the disease?

Example: $S = \{A\}$, $k = 2$, $r = 2$.



Results:

- Polytime solvable on trees.
- NP-hard and W[1]-hard wrt. k for lifetime two. Deligkas & Potapov [AAAI '20]
- W[1]-hard wrt. r for lifetime two.

Minimizing Reachability by Delaying

Minimizing Reachability by Delaying

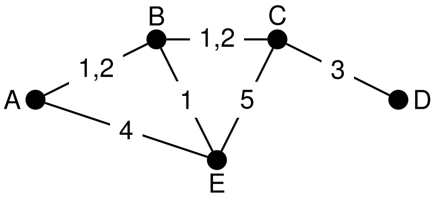
Given some initial outbreak S, can we **delay** up to k time-edges such that at most r vertices eventually contract the disease?

Minimizing Reachability by Delaying

Minimizing Reachability by Delaying

Given some initial outbreak S, can we **delay** up to k time-edges such that at most r vertices eventually contract the disease?

Example: $S = \{A\}$, $k = 2$, $r = 3$.

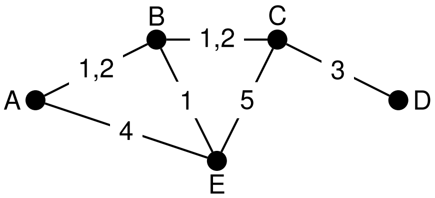


Minimizing Reachability by Delaying

Minimizing Reachability by Delaying

Given some initial outbreak S, can we **delay** up to k time-edges such that at most r vertices eventually contract the disease?

Example: $S = \{A\}$, $k = 2$, $r = 3$.



Results:

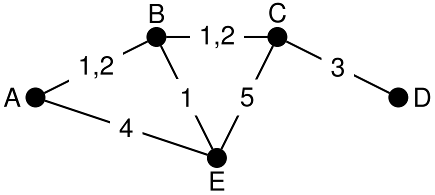
- Polytime solvable on trees.

Minimizing Reachability by Delaying

Minimizing Reachability by Delaying

Given some initial outbreak S, can we **delay** up to k time-edges such that at most r vertices eventually contract the disease?

Example: $S = \{A\}$, $k = 2$, $r = 3$.



Results:

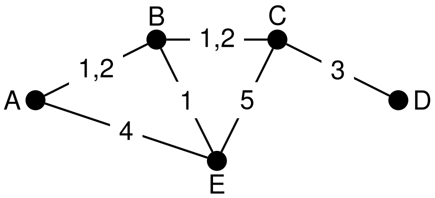
- Polytime solvable on trees.
- NP-hard and W[1]-hard wrt. k for lifetime two. Deligkas & Potapov [AAAI '20]

Minimizing Reachability by Delaying

Minimizing Reachability by Delaying

Given some initial outbreak S, can we **delay** up to k time-edges such that at most r vertices eventually contract the disease?

Example: $S = \{A\}$, $k = 2$, $r = 3$.



Results:

- Polytime solvable on trees.
- NP-hard and W[1]-hard wrt. k for lifetime two. Deligkas & Potapov [AAAI '20]
- Fixed-parameter tractable wrt. r.

FPT Algorithm for Delaying: Idea

Algorithm Idea:

1. Move to equivalent model:

   Instead of **delaying** time-edges, **slow** them.

   $\rightsquigarrow$ On the chosen time-edges, transmission takes 2 time steps instead of 1.

FPT Algorithm for Delaying: Idea

Algorithm Idea:

1. Move to equivalent model:

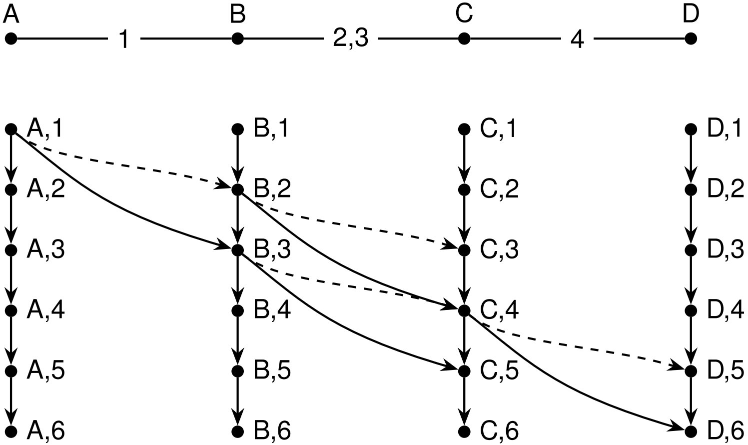
   Instead of **delaying** time-edges, **slow** them.

   $\rightsquigarrow$ On the chosen time-edges, transmission takes 2 time steps instead of 1.

2. If we are given a set R of infected vertices, we can transform to a flow problem.

FPT Algorithm for Delaying: Flow Network

FPT Algorithm for Delaying: Flow Network



FPT Algorithm for Delaying: Idea

Algorithm Idea:

1. Move to equivalent model:

   Instead of **delaying** time-edges, **slow** them.

   $\rightsquigarrow$ On the chosen time-edges, transmission takes 2 time steps instead of 1.

2. If we are given a set R of infected vertices, we can transform to a flow problem.

3. If maximum flow has value $\leq k$, then the corresponding min-cut is a solution!

FPT Algorithm for Delaying: Idea

Algorithm Idea:

1. Move to equivalent model:

   Instead of **delaying** time-edges, **slow** them.

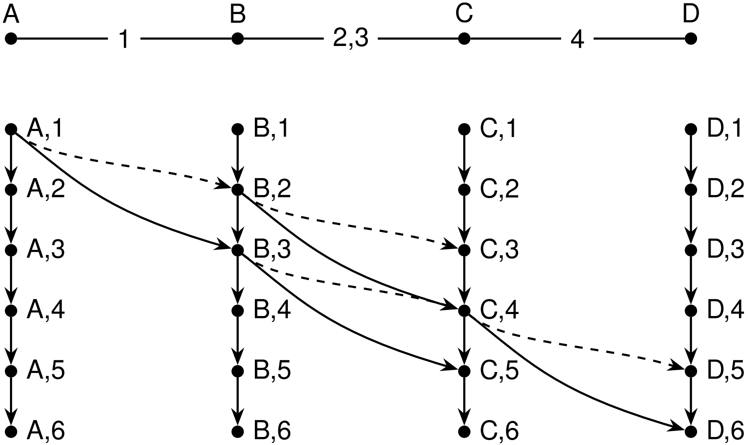
   $\rightsquigarrow$ On the chosen time-edges, transmission takes 2 time steps instead of 1.

2. If we are given a set R of infected vertices, we can transform to a flow problem.

3. If maximum flow has value $\leq k$, then the corresponding min-cut is a solution!

4. If maximum flow has value $> k$, consider set L of flow network vertices that “leak” flow to $V \setminus R$.

FPT Algorithm for Delaying: Flow Network



FPT Algorithm for Delaying: Idea

Algorithm Idea:

1. Move to equivalent model:

   Instead of **delaying** time-edges, **slow** them.

   $\rightsquigarrow$ On the chosen time-edges, transmission takes 2 time steps instead of 1.

2. If we are given a set R of infected vertices, we can transform to a flow problem.
3. If maximum flow has value $\leq k$, then the corresponding min-cut is a solution!
4. If maximum flow has value $> k$, consider set L of flow network vertices that "leak" flow to $V \setminus R$.

FPT Algorithm for Delaying: Idea

Algorithm Idea:

1. Move to equivalent model:
   Instead of **delaying** time-edges, **slow** them.
   $\rightsquigarrow$ On the chosen time-edges, transmission takes 2 time steps instead of 1.
2. If we are given a set R of infected vertices, we can transform to a flow problem.
3. If maximum flow has value $\leq k$, then the corresponding min-cut is a solution!
4. If maximum flow has value $> k$, consider set L of flow network vertices that "leak" flow to $V \setminus R$.
   In every solution, at least one $(v, t) \in L$ stays reachable!

FPT Algorithm for Delaying: Idea

Algorithm Idea:

1. Move to equivalent model:

   Instead of **delaying** time-edges, **slow** them.

   $\rightsquigarrow$ On the chosen time-edges, transmission takes 2 time steps instead of 1.

2. If we are given a set R of infected vertices, we can transform to a flow problem.

3. If maximum flow has value $\leq k$, then the corresponding min-cut is a solution!

4. If maximum flow has value $> k$, consider set L of flow network vertices that “leak” flow to $V \setminus R$.

   In every solution, at least one $(v, t) \in L$ stays reachable!

   $\rightsquigarrow$ Search-tree!

FPT Algorithm for Delaying: Search-Tree

Search-Tree Algorithm:

1. Start with $R \leftarrow S$ where S are the source vertices.

FPT Algorithm for Delaying: Search-Tree

Search-Tree Algorithm:

1. Start with $R \leftarrow S$ where S are the source vertices.
2. Transform to flow problem and check if there is a flow of value $> k$.

FPT Algorithm for Delaying: Search-Tree

Search-Tree Algorithm:

1. Start with $R \leftarrow S$ where S are the source vertices.
2. Transform to flow problem and check if there is a flow of value $> k$.
   - If no, then the corresponding min-cut is a solution.

FPT Algorithm for Delaying: Search-Tree

Search-Tree Algorithm:

1. Start with $R \leftarrow S$ where S are the source vertices.
2. Transform to flow problem and check if there is a flow of value $> k$.
   - If no, then the corresponding min-cut is a solution.
   - If yes, compute a “small” set of neighbors Y of “leaking” network vertices $(v, t) \in L$.

FPT Algorithm for Delaying: Search-Tree

Search-Tree Algorithm:

1. Start with $R \leftarrow S$ where S are the source vertices.
2. Transform to flow problem and check if there is a flow of value $> k$.
   - If no, then the corresponding min-cut is a solution.
   - If yes, compute a “small” set of neighbors Y of “leaking” network vertices $(v, t) \in L$.
   - Iterate over vertices $v \in Y$.
     - $R' \leftarrow R \cup \{v\}$.

FPT Algorithm for Delaying: Search-Tree

Search-Tree Algorithm:

1. Start with $R \leftarrow S$ where S are the source vertices.
2. Transform to flow problem and check if there is a flow of value $> k$.
   - If no, then the corresponding min-cut is a solution.
   - If yes, compute a “small” set of neighbors Y of “leaking” network vertices $(v, t) \in L$.
   - Iterate over vertices $v \in Y$.
     - $R' \leftarrow R \cup \{v\}$.
     - If $|R'| \leq r$ then recursively continue with R' at Step 2.

FPT Algorithm for Delaying: Search-Tree

Search-Tree Algorithm:

1. Start with $R \leftarrow S$ where S are the source vertices.
2. Transform to flow problem and check if there is a flow of value $> k$.
    - If no, then the corresponding min-cut is a solution.
    - If yes, compute a “small” set of neighbors Y of “leaking” network vertices $(v, t) \in L$.
    - Iterate over vertices $v \in Y$.
        - $R' \leftarrow R \cup \{v\}$.
        - If $|R'| \leq r$ then recursively continue with R' at Step 2.
        - Otherwise, output “no solution”.

FPT Algorithm for Delaying: Search-Tree

Search-Tree Algorithm:

1. Start with $R \leftarrow S$ where S are the source vertices.
2. Transform to flow problem and check if there is a flow of value $> k$.
   - If no, then the corresponding min-cut is a solution.
   - If yes, compute a "small" set of neighbors Y of "leaking" network vertices $(v, t) \in L$.
   - Iterate over vertices $v \in Y$.
     - $R' \leftarrow R \cup \{v\}$.
     - If $|R'| \leq r$ then recursively continue with R' at Step 2.
     - Otherwise, output "no solution".

Theorem

Minimizing Reachability by Delaying can be solved in $O(r^r \cdot k \cdot |G|)$ time.

Conclusion and Future Work

Summary:

- Reachability Minimization by Time-Edge Modification is a wide field with many still open problems.

Conclusion and Future Work

Summary:

- Reachability Minimization by Time-Edge Modification is a wide field with many still open problems.
- For the case where the “outbreak” is known, surprising difference in parameterized complexity wrt. reachable set size between “delete” and “delay” version.

Conclusion and Future Work

Summary:

- Reachability Minimization by Time-Edge Modification is a wide field with many still open problems.
- For the case where the “outbreak” is known, surprising difference in parameterized complexity wrt. reachable set size between “delete” and “delay” version.
- Both versions poly-time solvable on trees.

Conclusion and Future Work

Summary:

- Reachability Minimization by Time-Edge Modification is a wide field with many still open problems.
- For the case where the “outbreak” is known, surprising difference in parameterized complexity wrt. reachable set size between “delete” and “delay” version.
- Both versions poly-time solvable on trees.

Future work:

- Generalize tree-algorithm.

Conclusion and Future Work

Summary:

- Reachability Minimization by Time-Edge Modification is a wide field with many still open problems.
- For the case where the “outbreak” is known, surprising difference in parameterized complexity wrt. reachable set size between “delete” and “delay” version.
- Both versions poly-time solvable on trees.

Future work:

- Generalize tree-algorithm.

Arxiv link:

Thank you!