Resolving Sets in Temporal Graphs

Jan Bok[1,2], Antoine Dailly[1], Tuomo Lehtilä[1,3,4]

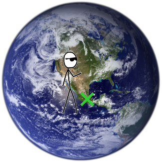
IWOCA, July 2nd, 2024

[1] LIMOS, Université Clermont-Auvergne, Clermont-Ferrand, France
[2] Faculty of Mathematics and Physics, Charles University, Prague
[3] Department of Computer Science, University of Helsinki, Helsinki, Finland
[4] Helsinki Institute for Information Technology (HIIT), Espoo, Finland

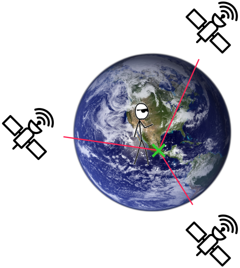
Funded by: ANR GRALMECO, I-SITE CAP 20-25, ERC Synergy Grant POCOCOP (GA 101071674), Business Finland Project 6GNTF

The problem we are studying

The problem we are studying

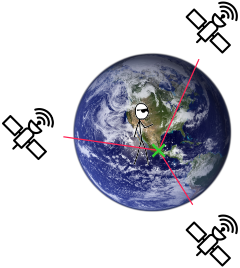


The problem we are studying



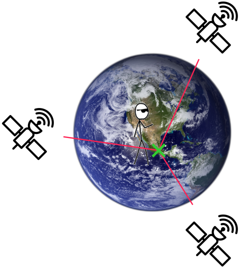
GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites

The problem we are studying

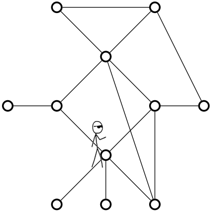


GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites

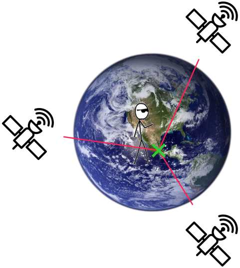
The problem we are studying



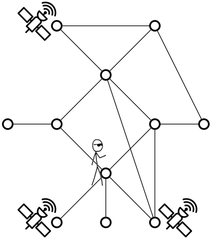
GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites



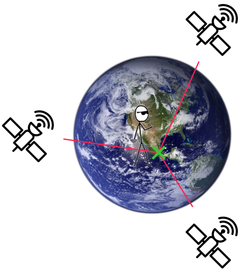
The problem we are studying



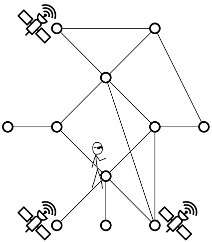
GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites



The problem we are studying

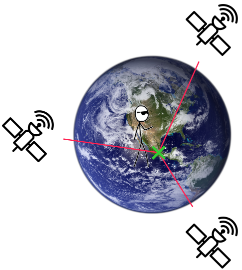


GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites

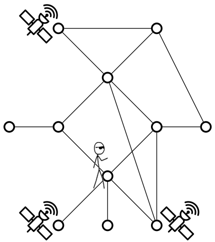


How many "satellites" would I need in a given graph?
⇒ Well-studied Metric Dimension

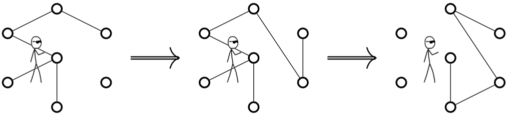
The problem we are studying



GPS, GLONASS, Galileo,
Beidou, IRNSS, QZSS: use
of at least four satellites



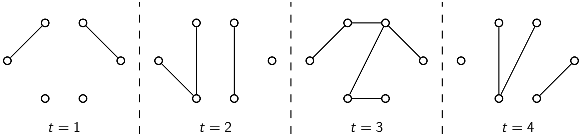
How many "satellites" would
I need in a given graph?
$\Rightarrow$ Well-studied Metric Dimension



What if the graph **changes over time**?

Temporal graphs

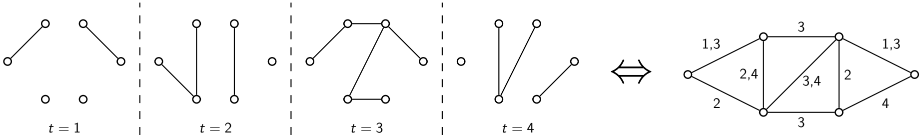
$$\mathcal{G} = (V, E_1, E_2, \ldots, E_{t_{\max}})$$ [Ferreira & Viennot, 2002]



Temporal graphs

$$\mathcal{G} = (V, E_1, E_2, \ldots, E_{t_{\max}}) \text{ [Ferreira \& Viennot, 2002]}$$

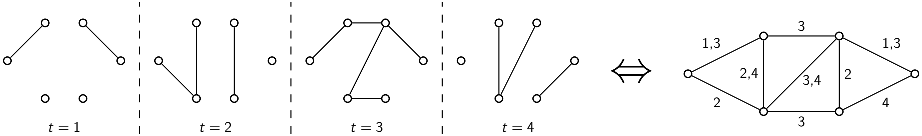
$$\mathcal{G} = (V, E, \lambda) \text{ [Kempe, Kleinberg \& Kumar, 2000]}$$



Temporal graphs

$\mathcal{G} = (V, E_1, E_2, \ldots, E_{\mathrm{t_{max}}})$ [Ferreira & Viennot, 2002]

$\mathcal{G} = (V, E, \lambda)$ [Kempe, Kleinberg & Kumar, 2000]



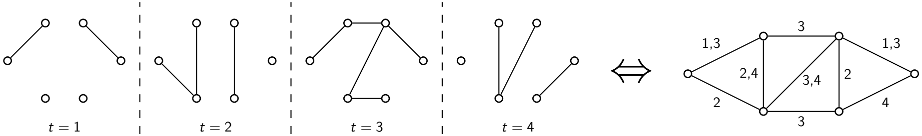
Well-studied in distributed algorithms and dynamic networks (biological, transportation...)

[Casteigts, 2018] for a thorough introduction

Temporal graphs

$$\mathcal{G} = (V, E_1, E_2, \ldots, E_{\mathrm{t_{max}}})$$ [Ferreira & Viennot, 2002]

$$\mathcal{G} = (V, E, \lambda)$$ [Kempe, Kleinberg & Kumar, 2000]



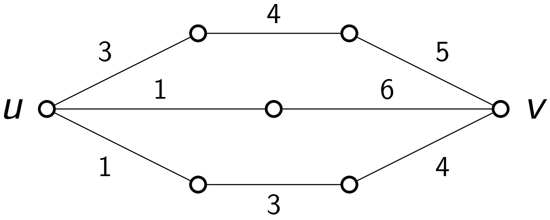
Well-studied in distributed algorithms and dynamic networks (biological, transportation...)
[Casteigts, 2018] for a thorough introduction

Useful terms

- The static graph $G = (V, E)$ is the **underlying graph**
- λ is the **time labeling**
- A **journey** is a path in the underlying graph with **strictly increasing** time-steps

Distance in temporal graphs

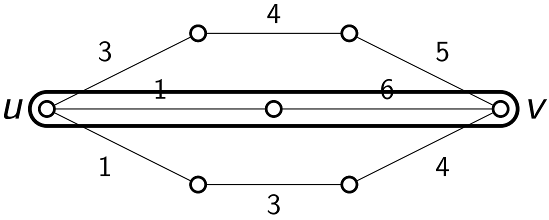
Three possible variables to minimize in a journey



Distance in temporal graphs

Three possible variables to minimize in a journey

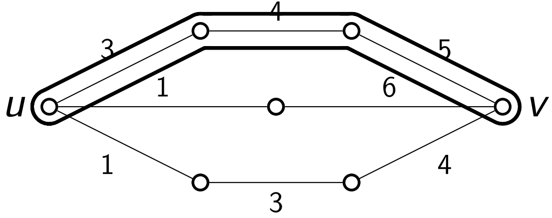
- **Shortest:** Number of edges in the underlying graph



Distance in temporal graphs

Three possible variables to minimize in a journey

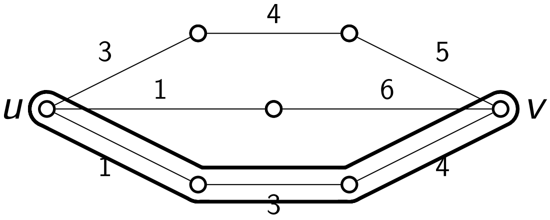
- **Shortest:** Number of edges in the underlying graph
- **Fastest:** Duration of the journey



Distance in temporal graphs

Three possible variables to minimize in a journey

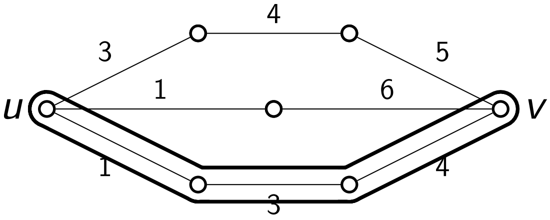
- **Shortest:** Number of edges in the underlying graph
- **Fastest:** Duration of the journey
- **Foremost:** Arrival time



Distance in temporal graphs

Three possible variables to minimize in a journey

- **Shortest:** Number of edges in the underlying graph
- **Fastest:** Duration of the journey
- **Foremost:** Arrival time



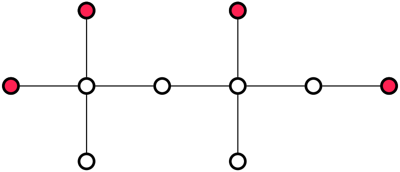
Temporal distance

The **temporal distance** from u to v is the last time-step of a **foremost** journey from u to v.

Resolving Set

Definition

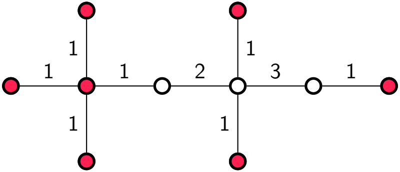
$R \subseteq V$ s.t., $\forall u, v \in V$, $u \neq v$, $\exists r \in R$ with $\text{dist}(r, u) \neq \text{dist}(r, v)$.



Temporal Resolving Set

Definition

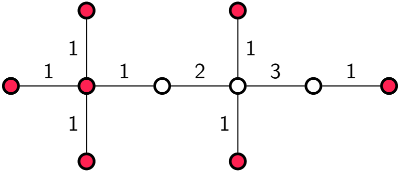
$R \subseteq V$ s.t., $\forall u, v \in V,\ u \neq v,\ \exists r \in R$ with $\mathbf{dist_t}(r, u) \neq \mathbf{dist_t}(r, v)$.



Temporal Resolving Set

Definition

$R \subseteq V$ s.t., $\forall u, v \in V$, $u \neq v$, $\exists r \in R$ with $\mathbf{dist_t}(r, u) \neq \mathbf{dist_t}(r, v)$.



- All vertices must be reached from some vertex of R
- If $\lambda(e) = \{1, \ldots, \text{diam}(G)\}$ for every edge e, then we get the standard resolving set $\rightarrow$ Generalization
- Temporal Resolving Set: find a minimum-size set

Previous work on resolving sets

Resolving set

- Defined by [Harary & Melter, 1976] and [Slater, 1975], well-studied since
- NP-hard, even on restricted classes (bipartite...)
- W[2]-hard (wrt solution size) on subcubic graphs
- Polynomial algorithms for trees and complete graphs

Previous work on resolving sets

Resolving set

- Defined by [Harary & Melter, 1976] and [Slater, 1975], well-studied since
- NP-hard, even on restricted classes (bipartite...)
- W[2]-hard (wrt solution size) on subcubic graphs
- Polynomial algorithms for trees and complete graphs

k-truncated resolving set

$\mathrm{dist}_k(u, v) = \min(\mathrm{dist}(u, v), k + 1)$

- Defined by [Estrada-Moreno *et al.*, 2021], generalizing adjacency resolving sets for any k
- NP-hard on trees, but XP (wrt k) algorithm
- Polynomial algorithms for subdivided stars and complete graphs

Our results: complexity of Temporal Resolving Set

	Standard resolving set	k-truncated resolving set	**Temporal resolving set**
Trees	poly	NP-hard XP wrt k	NP-hard (2 consecutive time labels per edge)
Subdivided stars	poly	poly	NP-hard (2 time labels per edge) poly (time labels 1 and 2, one per edge)
Stars, Paths	poly	poly	poly (1 time label per edge)
Complete graphs	poly	poly	NP-hard (time labels 1 and 2, one per edge)

NP-hardness

Theorem [B., Dailly & Lehtilä, 2024]

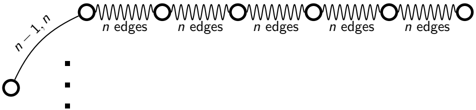
TEMPORAL RESOLVING SET is **NP-hard**, even if the underlying graph is a **tree** and each edge appears **at most twice**.

NP-hardness

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET is **NP-hard**, even if the underlying graph is a **tree** and each edge appears **at most twice**.

Proof idea: reduction from 3-DIMENSIONAL MATCHING

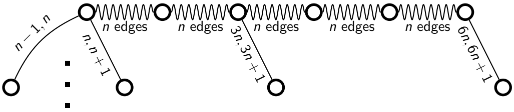


NP-hardness

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET is **NP-hard**, even if the underlying graph is a **tree** and each edge appears **at most twice**.

Proof idea: reduction from 3-DIMENSIONAL MATCHING



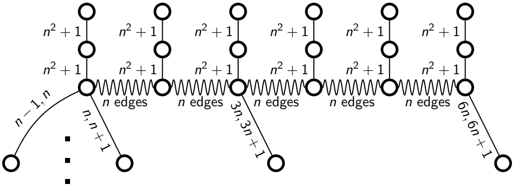
triple $(1, 3, 6)$

NP-hardness

Theorem [B., Dailly & Lehtilä, 2024]

Temporal Resolving Set is **NP-hard**, even if the underlying graph is a **tree** and each edge appears **at most twice**.

Proof idea: reduction from 3-Dimensional Matching



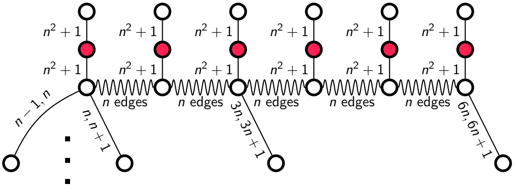
triple $(1, 3, 6)$

NP-hardness

Theorem [B., Dailly & Lehtilä, 2024]

Temporal Resolving Set is **NP-hard**, even if the underlying graph is a **tree** and each edge appears **at most twice**.

Proof idea: reduction from 3-Dimensional Matching



triple $(1,3,6)$

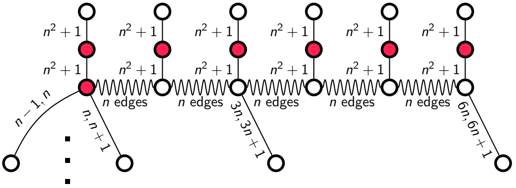
- Control vertices are always in the set

NP-hardness

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET is **NP-hard**, even if the underlying graph is a **tree** and each edge appears **at most twice**.

Proof idea: reduction from 3-DIMENSIONAL MATCHING



triple $(1, 3, 6)$
not in the matching

- Control vertices are always in the set
- First vertex in the set for every triple **not** in the matching

Paths

Theorem [B., Dailly & Lehtilä, 2024]

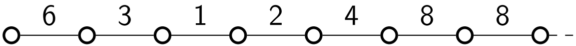
TEMPORAL RESOLVING SET can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Paths

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Algorithm idea

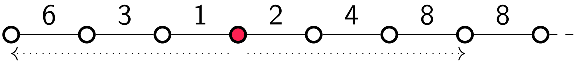


Paths

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Algorithm idea

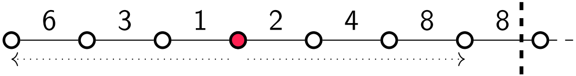


Paths

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Algorithm idea



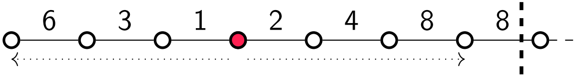
Iterate

Paths

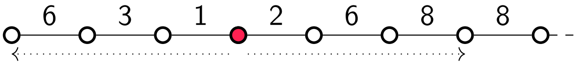
Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Algorithm idea



Iterate

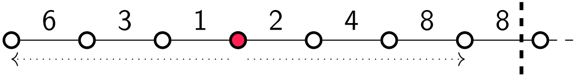


Paths

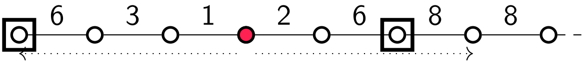
Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Algorithm idea



Iterate



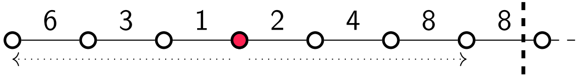
Conflict

Paths

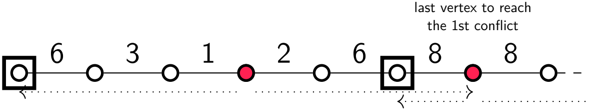
Theorem [B., Dailly & Lehtilä, 2024]

Temporal Resolving Set can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Algorithm idea



Iterate



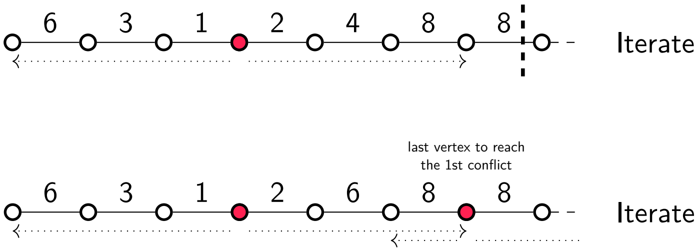
Conflict solved

Paths

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Algorithm idea

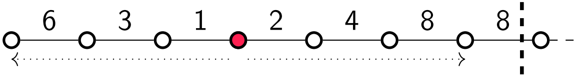


Paths

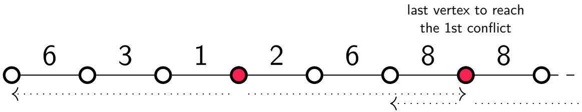
Theorem [B., Dailly & Lehtilä, 2024]

Temporal Resolving Set can be solved in **linear** time if the underlying graph is a **path** and each edge appears **once**.

Algorithm idea



Iterate



Iterate

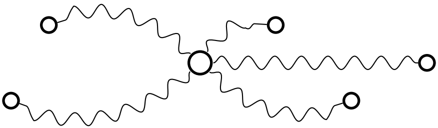
- A few more details for edge cases
- Proof of optimality: technical analysis

Subdivided stars

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in **polynomial** time if the underlying graph is a **subdivided star**, each edge appears **once** and the time-steps are **1 or 2**.

Algorithm idea

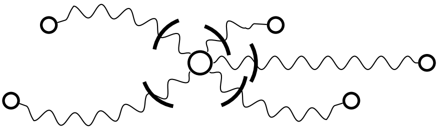


Subdivided stars

Theorem [B., Dailly & Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in **polynomial** time if the underlying graph is a **subdivided star**, each edge appears **once** and the time-steps are **1 or 2**.

Algorithm idea



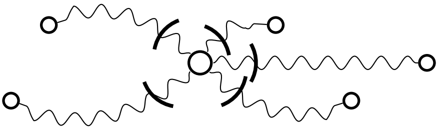
- Apply the path algorithm on every branch

Subdivided stars

Theorem [B., Dailly & Lehtilä, 2024]

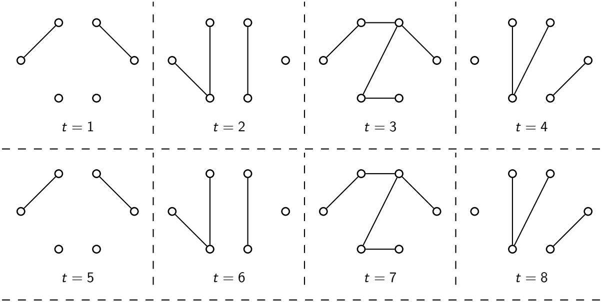
TEMPORAL RESOLVING SET can be solved in **polynomial** time if the underlying graph is a **subdivided star**, each edge appears **once** and the time-steps are **1 or 2**.

Algorithm idea



- Apply the path algorithm on every branch
- Manage the center and the link with the branches (lots of cases to consider carefully!)

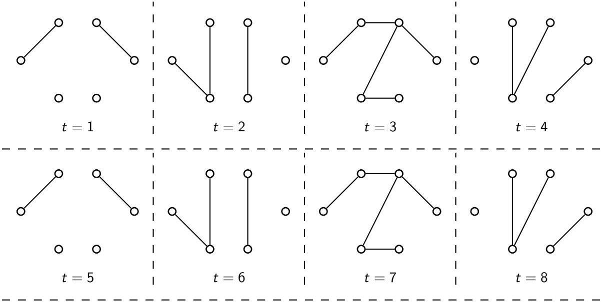
Our results: periodic time labelings



$\bullet \quad \bullet \quad \bullet$

$\rightarrow$ 4-periodic time labeling

Our results: periodic time labelings



• • •

$\rightarrow$ 4-periodic time labeling

Results for 1 time label per edge

- Combinatorial results (bounds) for paths, cycles, complete graphs, subdivided stars, and complete binary trees
- XP algorithm (wrt **period**) for subdivided stars

Periodic time labelings on subdivided stars

Periodic time labelings on subdivided stars

Proposition [B. Dailly, Lehtilä, 2024]

Any leaf is a temporal resolving set of a path with a periodic time labeling.

Periodic time labelings on subdivided stars

Proposition [B. Dailly, Lehtilä, 2024]

Any leaf is a temporal resolving set of a path with a periodic time labeling.

Theorem [B. Dailly, Lehtilä, 2024]

TEMPORAL RESOLVING SET can be solved in time $\mathcal{O}(n^{p+1})$ on a **subdivided star** of order **n** if the time labeling is **p-periodic** and each edge appears **once in every period**.

Periodic time labelings on subdivided stars

Proposition [B. Dailly, Lehtilä, 2024]

Any leaf is a temporal resolving set of a path with a periodic time labeling.

Theorem [B. Dailly, Lehtilä, 2024]

Temporal Resolving Set can be solved in time $\mathcal{O}(n^{p+1})$ on a **subdivided star** of order **n** if the time labeling is **p-periodic** and each edge appears **once in every period**.

Proof idea

- Lemma: for a tree in this setting, there is a minimum-size temporal resolving set containing only leaves
- At least $\ell - p$ leaves are necessary ($\ell =$ number of leaves)
- Test every set of $\ell - p$, $\ell - p + 1$, ..., ℓ leaves

Final words

Our contributions

- Introducing temporal resolving sets
- NP-hard even on subdivided stars and complete graphs
- Poly-time algorithms for paths, stars (under constraints)
- XP on subdivided stars for periodic labelings (wrt period)

Final words

Our contributions

- Introducing temporal resolving sets
- NP-hard even on subdivided stars and complete graphs
- Poly-time algorithms for paths, stars (under constraints)
- XP on subdivided stars for periodic labelings (wrt period)

Future work

- Other parameters: number of time-steps, ...
- Large number of labels per edge might help!
- Other notions of distance (shortest and fastest journey)

Final words

Our contributions

- Introducing temporal resolving sets
- NP-hard even on subdivided stars and complete graphs
- Poly-time algorithms for paths, stars (under constraints)
- XP on subdivided stars for periodic labelings (wrt period)

Future work

- Other parameters: number of time-steps, ...
- Large number of labels per edge might help!
- Other notions of distance (shortest and fastest journey)