Temporal Graphs: Algorithms and Complexity

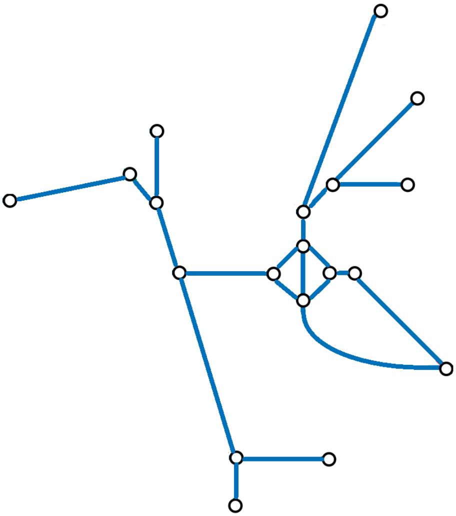
Eleni C. Akrida

Department of Computer Science,
Durham University, UK

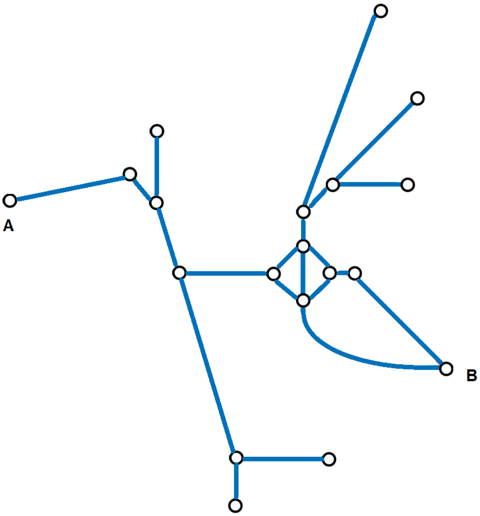
37th British Colloquium for Theoretical Computer Science

March 2021

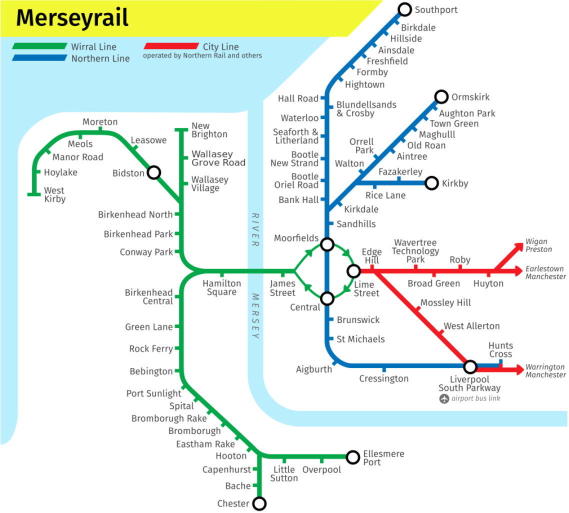
Static and Temporal Graphs



Static and Temporal Graphs



Static and Temporal Graphs



Static and Temporal Graphs

Many systems in Science and Technology:

- abstracted as graphs
- vertex $\longleftrightarrow$ elementary system unit
- edge $\longleftrightarrow$ some kind of interaction between units

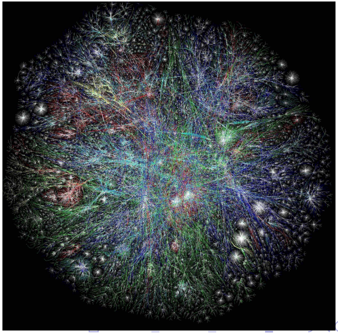
Static and Temporal Graphs

Many systems in Science and Technology:

- abstracted as graphs
- vertex $\longleftrightarrow$ elementary system unit
- edge $\longleftrightarrow$ some kind of interaction between units

However many modern systems are highly dynamic:

- Modern communication networks, e.g., mobile ad hoc, sensor, peer-to-peer, opportunistic, delay-tolerant networks: links change dynamically at a high rate
- Social networks: friendships are added/removed, individuals leave, new ones enter
- Physical systems: e.g. systems of interacting particles
- Transportation networks: transportation units change with time their position in the network



Static and Temporal Graphs

Network changes may:

- follow specific patterns, e.g. satellites following a trajectory, or
- be unpredictable, e.g. mobile ad hoc networks

Static and Temporal Graphs

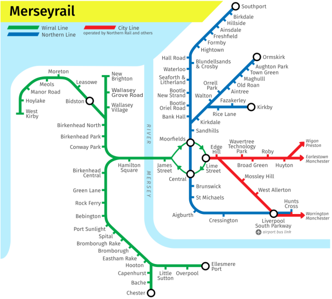
Network changes may:

- follow specific patterns, e.g. satellites following a trajectory, or
- be unpredictable, e.g. mobile ad hoc networks

The common characteristic in all these applications:

- the graph topology is subject to discrete changes over time

$\Rightarrow$ the notion of vertex adjacency must be appropriately re-defined
(by introducing the time dimension in the graph definition)



Various graph concepts (e.g. reachability, connectivity):

- crucially depend on the exact temporal ordering of the edges

Overview

- Temporal graphs
- Temporal paths (journeys)
- Strongly connected components
- Temporal exploration
- Temporal design problems
- Stochastic temporal graphs
- Future research directions

Temporal graphs

Formally:

Definition (Temporal Graph)

A temporal graph is a pair (G, λ) where:

- $G = (V, E)$ is an underlying (di)graph and
- $\lambda : E \to 2^{\mathbb{N}}$ is a discrete time-labeling function.

Temporal graphs

Formally:

Definition (Temporal Graph)

A temporal graph is a pair (G, λ) where:

- $G = (V, E)$ is an underlying (di)graph and
- $\lambda : E \to 2^{\mathbb{N}}$ is a discrete time-labeling function.

- If $t \in \lambda(e)$ then edge e is available at time t
- This formal definition (for single-availabilities per edge) embarks from:
  [Kempe, Kleinberg, Kumar, *STOC*, 2000]
  [Berman, *Networks*, 1996]
- In general every edge can have multiple availabilities
  [Mertzios, Michail, Chatzigiannakis, Spirakis, *ICALP*, 2013]

Temporal graphs

Formally:

Definition (Temporal Graph)

A temporal graph is a pair (G, λ) where:

- $G = (V, E)$ is an underlying (di)graph and
- $\lambda : E \to 2^{\mathbb{N}}$ is a discrete time-labeling function.

Remarks:

- Denote by λ_{min} (resp. λ_{max}) the smallest (resp. largest) time-label in (G, λ)
  - λ_{max} may be infinite (e.g. in periodic temporal graphs)
  - If $\lambda_{\text{max}} \neq \infty$, then the age of (G, λ) is $\alpha(\lambda) = \lambda_{\text{max}} - \lambda_{\text{min}} + 1$
- Unless otherwise specified:
  - the labels are given explicitly with the input
  - $c(\lambda)$ is the total number of labels

Temporal graphs

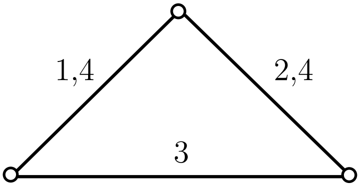
Formally:

Definition (Temporal Graph)

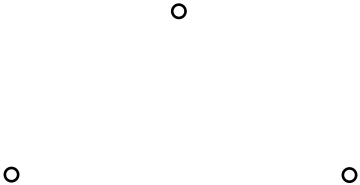
A temporal graph is a pair (G, λ) where:

- $G = (V, E)$ is an underlying (di)graph and
- $\lambda : E \to 2^{\mathbb{N}}$ is a discrete time-labeling function.

temporal graph:



temporal instances:



Temporal graphs

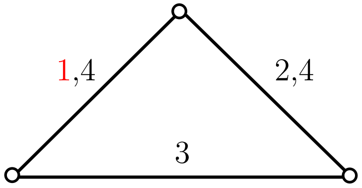
Formally:

Definition (Temporal Graph)

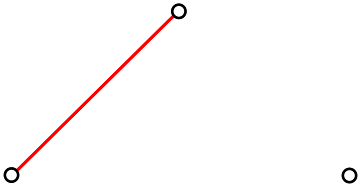
A temporal graph is a pair (G, λ) where:

- $G = (V, E)$ is an underlying (di)graph and
- $\lambda : E \to 2^{\mathbb{N}}$ is a discrete time-labeling function.

temporal graph:



temporal instances:



Temporal graphs

Formally:

Definition (Temporal Graph)

A temporal graph is a pair (G, λ) where:

- $G = (V, E)$ is an underlying (di)graph and
- $\lambda : E \to 2^{\mathbb{N}}$ is a discrete time-labeling function.

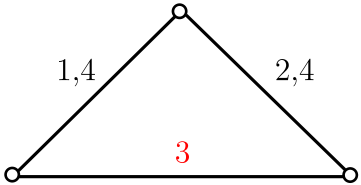
Formally:

Definition (Temporal Graph)

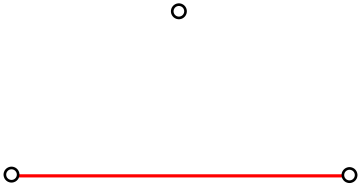
A temporal graph is a pair (G, λ) where:

- $G = (V, E)$ is an underlying (di)graph and
- $\lambda : E \to 2^{\mathbb{N}}$ is a discrete time-labeling function.

temporal graph:



temporal instances:



Temporal graphs

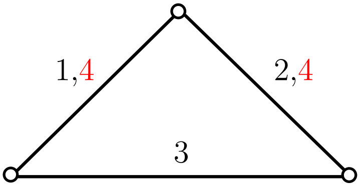
Formally:

Definition (Temporal Graph)

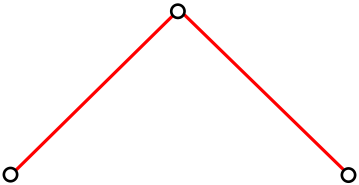
A temporal graph is a pair (G, λ) where:

- $G = (V, E)$ is an underlying (di)graph and
- $\lambda : E \to 2^{\mathbb{N}}$ is a discrete time-labeling function.

temporal graph:



temporal instances:



Related notions of dynamicity in graphs:

- flows over time
  [Fleischer, Skutella, *SIAM J. on Computing*, 2007]
  [Hoppe, Tardos, *Math. Oper. Res.*, 2000]
  [Fleischer, Tardos, *Oper. Res. Lett.*, 1998]
  - flows on static graph topologies with transit times on the edges
  - continuous availabilities; natural model, different techniques

Related notions of dynamicity in graphs:

- flows over time
  [Fleischer, Skutella, *SIAM J. on Computing*, 2007]
  [Hoppe, Tardos, *Math. Oper. Res.*, 2000]
  [Fleischer, Tardos, *Oper. Res. Lett.*, 1998]
    - flows on static graph topologies with transit times on the edges
    - continuous availabilities; natural model, different techniques
- minimum label graph problems
  [Fellows, Guo, Kanj, *J. Comp. Syst. Sci.*, 2010]
    - input: static topology G with a label on each edge, graph property Π
    - goal: find an edge subset with the smallest number of distinct labels which satisfies Π

Related notions of dynamicity in graphs:

- dynamic graphs
  [Demetrescu, Finocchi, Italiano, *Handbook Data Str. and Appl.*, 2004]
    - topology changes via insertion/deletion of vertices/edges
    - changes are assumed to happen rarely
    - goals: efficient query & solution update after a dynamic change

Related notions of dynamicity in graphs:

- dynamic graphs
  [Demetrescu, Finocchi, Italiano, *Handbook Data Str. and Appl.*, 2004]
  - topology changes via insertion/deletion of vertices/edges
  - changes are assumed to happen rarely
  - goals: efficient query & solution update after a dynamic change

In contrast, in the context of temporal networks:

- topology is expected to change frequently and massively
  $\Rightarrow$ changes are not anomalies or exceptions
- they are rather an integral part of the system
  $\Rightarrow$ can not be reasonably modeled with network faults /failures

Temporal graphs

Temporal graphs were studied under various different names:

- time-varying graphs
  [Aaron et al., *WG*, 2014]
  [Flocchini et al., *ISAAC*, 2009]
  [Tang et al., *ACM Comp. Comm. Review*, 2010]
- evolving graphs (usually "graph-centric")
  [Avin et al., *ICALP*, 2008]
  [Clementi et al., *SIAM J. Discr. Math.*, 2010]
  [Ferreira, *IEEE Network*, 2004]
- dynamic graphs
  [Giakkoupis et al., *ICALP*, 2014]
  [Casteigts et al., *Int. J. Par., Emergent & Distr. Syst*, 2012]
  [Bhadra and Ferreira, *J. Internet Serv. Appl.*, 2012]
- graphs over time
  [Leskovec et al., *ACM Trans. Knowl. Disc. from Data*, 2007]

Temporal graphs

Recent surveys and books:

- Time-Varying Graphs and Dynamic Networks
  [Casteigts et al., *Int. J. Par., Emergent & Distr. Syst*, 2012]
  - an attempt to integrate and unify existing models and concepts
- Deterministic Algorithms in Dynamic Networks
  [Casteigts, Flocchini, *Defence R&D Canada, Tech. Report I*, 2013]
  [Casteigts, Flocchini, *Defence R&D Canada, Tech. Report II*, 2013]
  - survey of deterministic algorithms for distributed computing
  - temporal graph classes based on temporal patterns of the labels

Temporal graphs

Recent surveys and books:

- Time-Varying Graphs and Dynamic Networks
  [Casteigts et al., *Int. J. Par., Emergent & Distr. Syst*, 2012]
  - an attempt to integrate and unify existing models and concepts
- Deterministic Algorithms in Dynamic Networks
  [Casteigts, Flocchini, *Defence R&D Canada, Tech. Report I*, 2013]
  [Casteigts, Flocchini, *Defence R&D Canada, Tech. Report II*, 2013]
  - survey of deterministic algorithms for distributed computing
  - temporal graph classes based on temporal patterns of the labels
    - satellites $\longrightarrow$ periodic availabilities
    - sensor networks $\longrightarrow$ connected at every instant
    - contacts in a company $\longrightarrow$ bounded edge recurrence (every week)
    - community contacts $\longrightarrow$ unbounded, yet recurrent interactions

Temporal graphs

Recent surveys and books:

- Time-Varying Graphs and Dynamic Networks
  [Casteigts et al., *Int. J. Par., Emergent & Distr. Syst*, 2012]
  - an attempt to integrate and unify existing models and concepts
- Deterministic Algorithms in Dynamic Networks
  [Casteigts, Flocchini, *Defence R&D Canada, Tech. Report I*, 2013]
  [Casteigts, Flocchini, *Defence R&D Canada, Tech. Report II*, 2013]
  - survey of deterministic algorithms for distributed computing
  - temporal graph classes based on temporal patterns of the labels
    - satellites $\longrightarrow$ periodic availabilities
    - sensor networks $\longrightarrow$ connected at every instant
    - contacts in a company $\longrightarrow$ bounded edge recurrence (every week)
    - community contacts $\longrightarrow$ unbounded, yet recurrent interactions
- Temporal Networks
  [Holme, Saramäki, eds., *Springer*, 2013]
  [Michail, Spirakis, *Commun. ACM*, 2018]

Overview

- Temporal graphs
- Temporal paths
- Strongly connected components
- Temporal exploration
- Temporal design problems
- Stochastic temporal graphs
- Future research directions

Temporal paths

The conceptual shift from static to temporal graphs significantly impacts:

- the definition of basic graph parameters
- the type of tasks to be computed

Graph properties can be classified as:

- a-temporal, i.e. satisfied at every instance
  - connectivity at every point in time
- temporal, i.e. satisfied over time
  - communication routes over time

Temporal paths

Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

Temporal paths

Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

Causality in information dissemination:

- information "flows" along edges whose labels respect time ordering
  $\Rightarrow$ strictly increasing labels along the path
- a "static path" given "in pieces"

Temporal paths

Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

Causality in information dissemination:

- information "flows" along edges whose labels respect time ordering
  $\Rightarrow$ strictly increasing labels along the path
- a "static path" given "in pieces"

Most identified temporal graph parameters are "temporal path"-related:

- temporal versions of distance, diameter, connectivity, reachability, exploration, etc.

Temporal paths

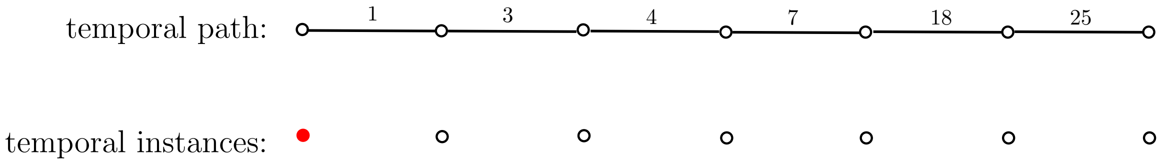
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A temporal path:



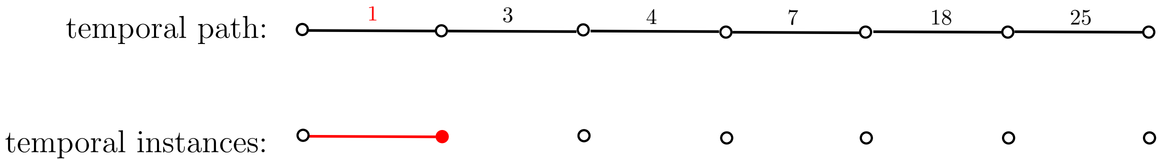
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A temporal path:



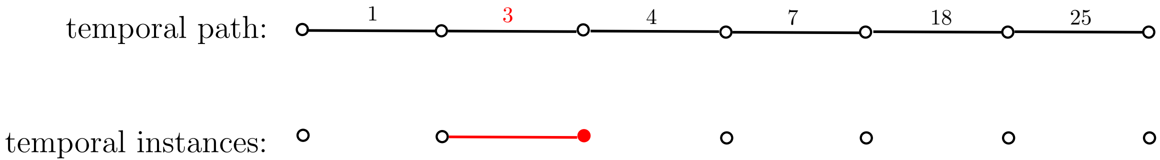
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \dots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \dots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \dots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A temporal path:



Temporal paths

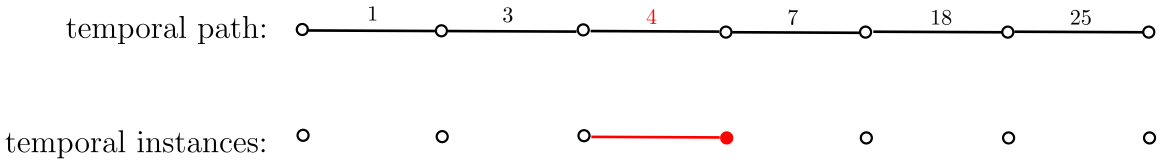
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A temporal path:



Temporal paths

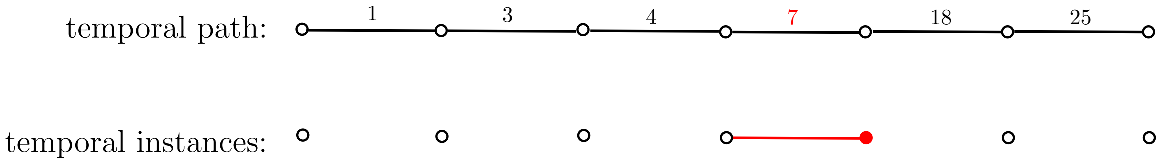
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A temporal path:



Temporal paths

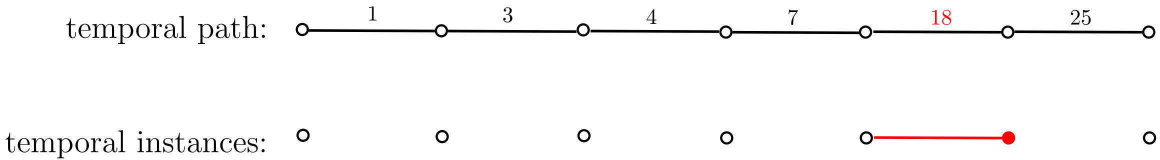
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A temporal path:



Temporal paths

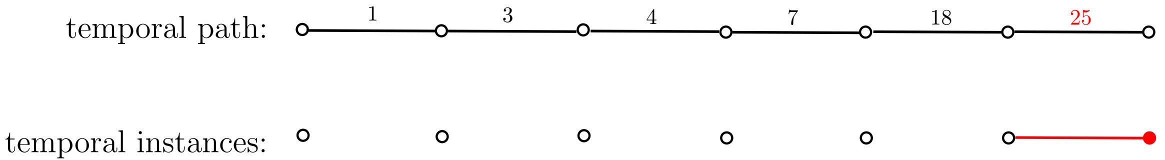
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A temporal path:



Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

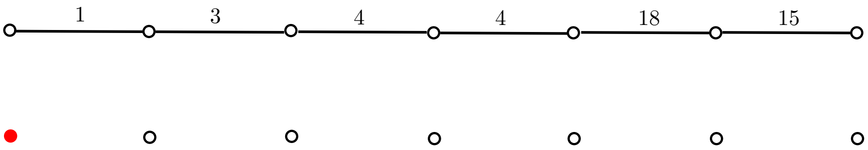
$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A non-temporal path:

non-temporal path:

temporal instances:



Temporal paths

Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

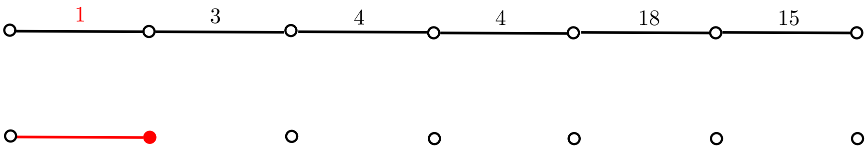
$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A non-temporal path:

non-temporal path:

temporal instances:



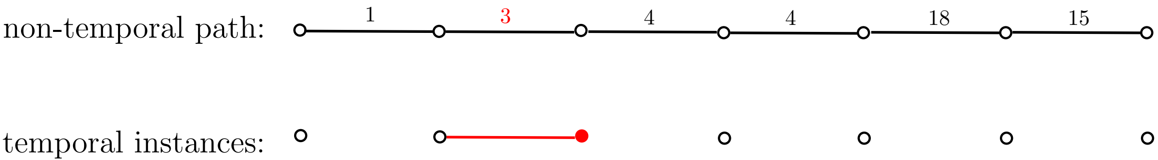
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A non-temporal path:



Temporal paths

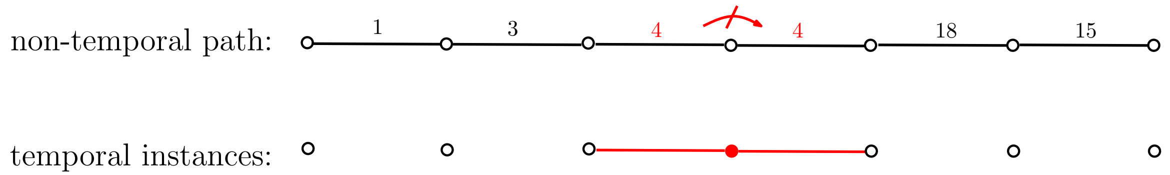
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A non-temporal path:



Temporal paths

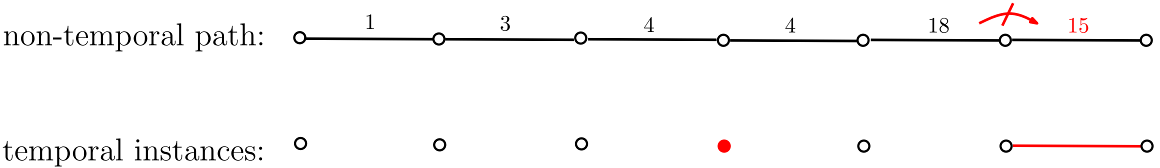
Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \ldots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \ldots, (e_k, \ell_k))$, where:

$$\ell_1 < \ell_2 < \ldots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A non-temporal path:



Temporal paths

Definition (Temporal path; Time-respecting path; Journey)

Let (G, λ) be a temporal graph and $P = (e_1, e_2, \dots, e_k)$ be a walk in G. A temporal path is a sequence $((e_1, \ell_1), (e_2, \ell_2), \dots, (e_k, \ell_k))$, where:

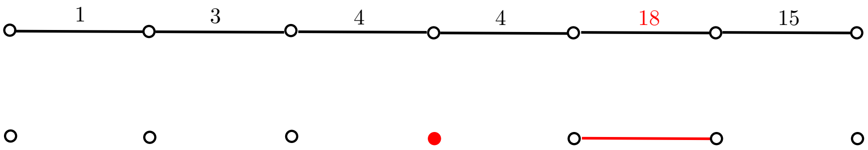
$$\ell_1 < \ell_2 < \dots < \ell_k$$

and $\ell_i \in \lambda(e_i)$, $1 \leq i \leq k$.

- A non-temporal path:

non-temporal path:

temporal instances:



Metrics to optimize

Question: What is the temporal analogue of an s-t shortest path?

Metrics to optimize

Question: What is the temporal analogue of an s-t shortest path?

Answer: Not uniquely defined!

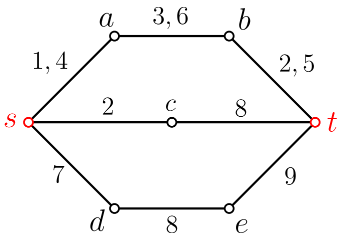
- topologically shortest path: smallest number of edges
- fastest path: smallest duration
- foremost path: smallest arrival time

Question: What is the temporal analogue of an s-t shortest path?

Answer: Not uniquely defined!

- topologically shortest path: smallest number of edges
- fastest path: smallest duration
- foremost path: smallest arrival time

Example:



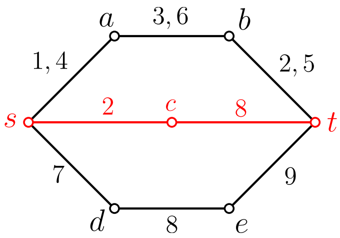
Metrics to optimize

Question: What is the temporal analogue of an s-t shortest path?

Answer: Not uniquely defined!

- topologically shortest path: smallest number of edges
- fastest path: smallest duration
- foremost path: smallest arrival time

Example:



shortest: s-c-t (two edges)

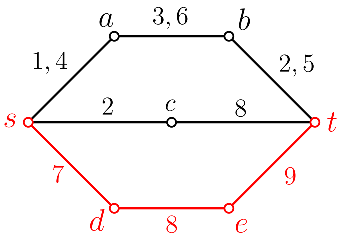
Metrics to optimize

Question: What is the temporal analogue of an s-t shortest path?

Answer: Not uniquely defined!

- topologically shortest path: smallest number of edges
- fastest path: smallest duration
- foremost path: smallest arrival time

Example:



shortest: s-c-t (two edges)

fastest: s-d-e-t (no intermediate waiting)

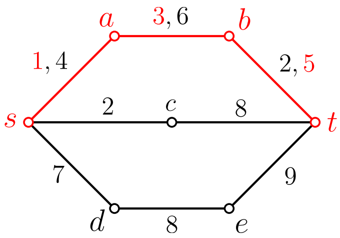
Metrics to optimize

Question: What is the temporal analogue of an s-t shortest path?

Answer: Not uniquely defined!

- topologically shortest path: smallest number of edges
- fastest path: smallest duration
- foremost path: smallest arrival time

Example:



shortest: s-c-t (two edges)

fastest: s-d-e-t (no intermediate waiting)

foremost: $s - a - b - t$ (arriving at time 5)

Metrics to optimize

An easy algorithm for computing all foremost paths from a given source s:

[Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017]

- first sort the time-labels non-decreasingly
- run a BFS-like search starting from s
- at every time-step t consider only edges currently available
- if you reach a new vertex at time t, keep its predecessor

An easy algorithm for computing all foremost paths from a given source s:

[Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017]

Algorithm 1 Foremost Temporal Paths from Source s

Let S be the array with the sorted time-labels
$R \leftarrow \{s\}$
for each $v \in V \setminus \{s\}$ **do**
    $pred[v] \leftarrow \emptyset$; $arr[v] \leftarrow \infty$ {Init.: Predecessor; Time Arrived}
for each time-label $t \in S$ **do**
    for each edge $e = (u, v)$ with $t \in \lambda(e)$ **do**
        if $u \in R$, $v \notin R$, and $arr[u] < t$ **then** {we reached v}
            $pred[v] \leftarrow u$; $arr[v] \leftarrow t$ {Predecessor; Time Arrived}
            $R \leftarrow R \cup \{v\}$

Metrics to optimize

An easy algorithm for computing all foremost paths from a given source s:

- easy adaptation of the static BFS algorithm
- running time $O(c(\lambda) \cdot \log(c(\lambda)))$
- due to the sorting of the labels

Algorithm 1 Foremost Temporal Paths from Source s

Let S be the array with the sorted time-labels
$R \leftarrow \{s\}$
for each $v \in V \setminus \{s\}$ **do**
  $pred[v] \leftarrow \emptyset$; $arr[v] \leftarrow \infty$ {Init.: Predecessor; Time Arrived}
for each time-label $t \in S$ **do**
  for each edge $e = (u, v)$ with $t \in \lambda(e)$ **do**
    if $u \in R$, $v \notin R$, and $arr[u] < t$ **then** {we reached v}
      $pred[v] \leftarrow u$; $arr[v] \leftarrow t$ {Predecessor; Time Arrived}
      $R \leftarrow R \cup \{v\}$

Metrics to optimize

Polynomial algorithms exist also in the case of edges with traversal times for computing:

- shortest and foremost paths   [adaptations of Dijkstra's algorithm]
- fastest paths

[Bui-Xuan, Ferreira, Jarry, *Int. J. Found. Comp. Sci.*, 2003]

Metrics to optimize

Polynomial algorithms exist also in the case of edges with traversal times for computing:

- shortest and foremost paths [adaptations of Dijkstra's algorithm]
- fastest paths

[Bui-Xuan, Ferreira, Jarry, *Int. J. Found. Comp. Sci.*, 2003]

Question: Are all "path-related" temporal problems tractable?

Metrics to optimize

Polynomial algorithms exist also in the case of edges with traversal times for computing:

- shortest and foremost paths [adaptations of Dijkstra's algorithm]
- fastest paths

[Bui-Xuan, Ferreira, Jarry, *Int. J. Found. Comp. Sci.*, 2003]

Question: Are all "path-related" temporal problems tractable?

Answer: Not all!

E.g. some temporal variations of:

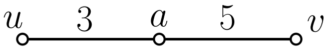
- connectivity problems
- reachability problems

Overview

- Temporal graphs
- Temporal paths
- Strongly connected components
- Temporal exploration
- Temporal design problems
- Stochastic temporal graphs
- Future research directions

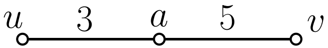
Temporal strongly connected components

- We write $u \rightsquigarrow v$ if there exists a temporal path from u to v
- The relation $\rightsquigarrow$ is not symmetric: $u \rightsquigarrow v \;\not\Leftrightarrow\; v \rightsquigarrow u$

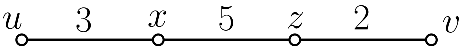


Temporal strongly connected components

- We write $u \rightsquigarrow v$ if there exists a temporal path from u to v
- The relation $\rightsquigarrow$ is not symmetric: $u \rightsquigarrow v \not\Leftrightarrow v \rightsquigarrow u$



- and not transitive: $u \rightsquigarrow z,\ z \rightsquigarrow v \not\Rightarrow u \rightsquigarrow v$



$\Rightarrow$ the time dimension creates its own "level of direction"

Temporal strongly connected components

Recall:

Definition

A directed (static) graph G is strongly connected if there is a path in each direction between each pair of vertices of G.

Temporal strongly connected components

Recall:

Definition

A directed (static) graph G is strongly connected if there is a path in each direction between each pair of vertices of G.

A key property:

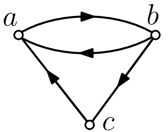
Observation

Let S be a (maximal) strongly connected subgraph and $u, v \in S$. If $P = (u, \dots, z, \dots, v)$ is a path from u to v then $z \in S$.

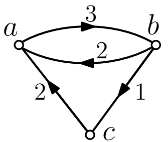
- Does this transfer to temporal graphs?

Temporal strongly connected components

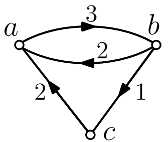
static:



temporal:

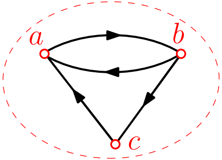


temporal:



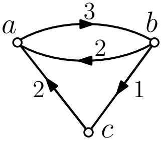
Temporal strongly connected components

static:

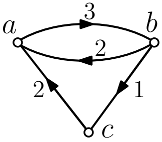


strongly
connected

temporal:

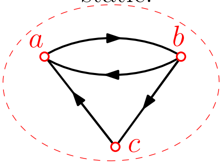


temporal:



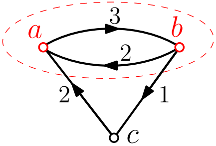
Temporal strongly connected components

static:



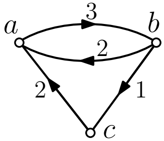
strongly connected

temporal:



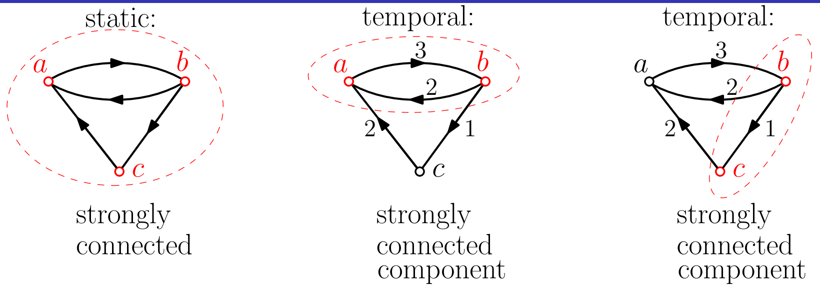
strongly connected component

temporal:



- $\{a, b\}$: direct temporal paths between a and b

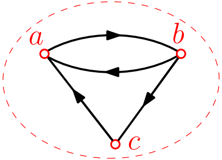
Temporal strongly connected components



- $\{a, b\}$: direct temporal paths between a and b
- $\{b, c\}$: the only temporal path from c to b passes through $a \notin \{b, c\}$

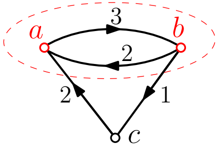
Temporal strongly connected components

static:



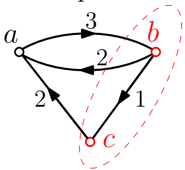
strongly
connected

temporal:



strongly
connected
component

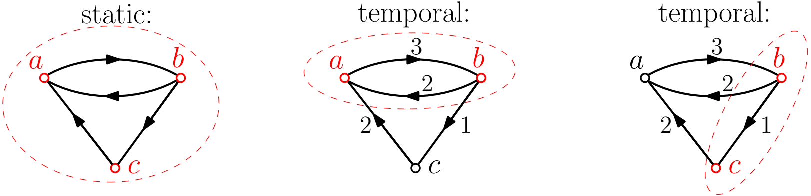
temporal:



strongly
connected
component

- $\{a, b\}$: direct temporal paths between a and b
- $\{b, c\}$: the only temporal path from c to b passes through $a \notin \{b, c\}$
- $\{a, b, c\}$: no temporal path from a to c

Temporal strongly connected components

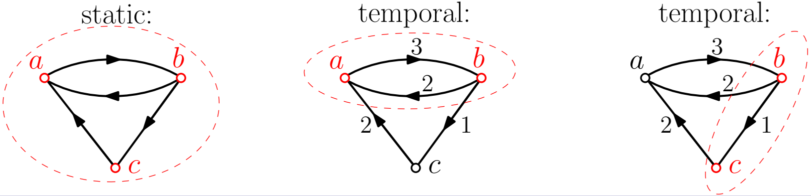


Definition (Bhadra, Ferreira, 2012)

An open strongly connected component (o-SCC) in a temporal graph is a set S of vertices such that $u \rightsquigarrow v$ for every $u, v \in S$.

Examples of an o-SCC: $\{a, b\}$, $\{b, c\}$

Temporal strongly connected components



Definition (Bhadra, Ferreira, 2012)

An open strongly connected component (o-SCC) in a temporal graph is a set S of vertices such that $u \rightsquigarrow v$ for every $u, v \in S$.

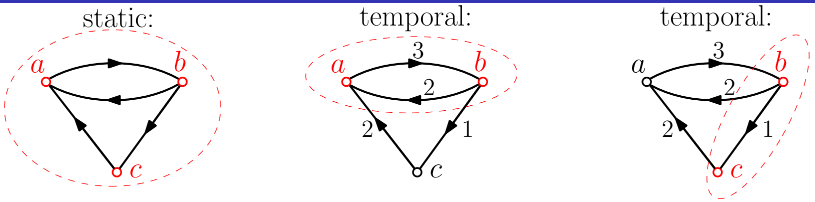
Examples of an o-SCC: $\{a, b\}$, $\{b, c\}$

Definition (Bhadra, Ferreira, 2012)

A strongly connected component (SCC) in a temporal graph is a set S of vertices such that, for every $u, v \in S$, there is a temporal path from u to v that uses only vertices from S.

Example of a SCC: $\{a, b\}$

Temporal strongly connected components



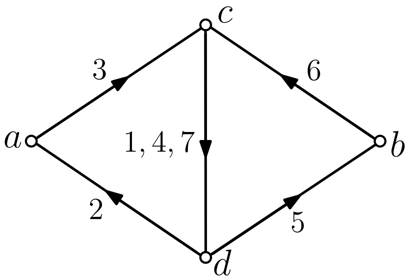
A difference to the static case:

- there can be a path between two vertices of the SCC (e.g. $\{a, b\}$) that traverses vertices outside the SCC (e.g. c)
- the same for an o-SCC (e.g. $\{b, c\}$)

Temporal strongly connected components

Further differences to the static case:

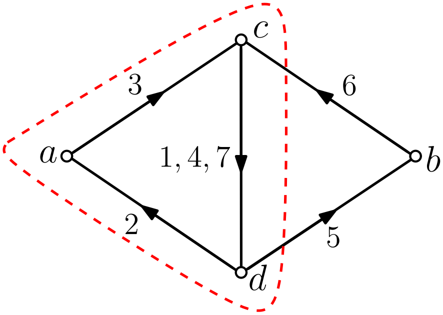
- two different SCCs can have common vertices



Temporal strongly connected components

Further differences to the static case:

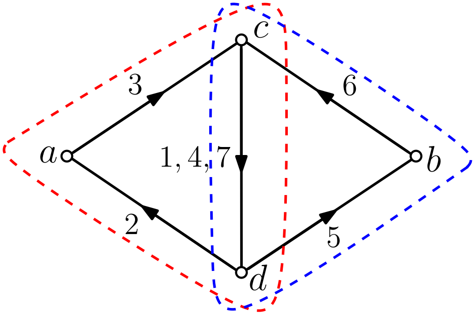
- two different SCCs can have common vertices
  - $\{a, c, d\}$ is a SCC



Temporal strongly connected components

Further differences to the static case:

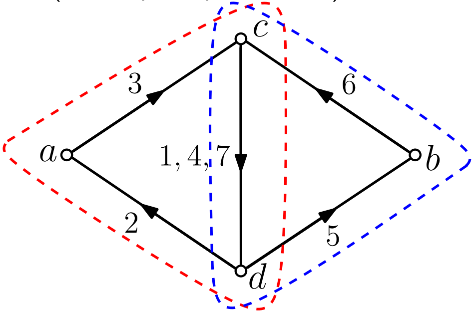
- two different SCCs can have common vertices
  - $\{a, c, d\}$ is a SCC
  - $\{b, c, d\}$ is another SCC



Temporal strongly connected components

Further differences to the static case:

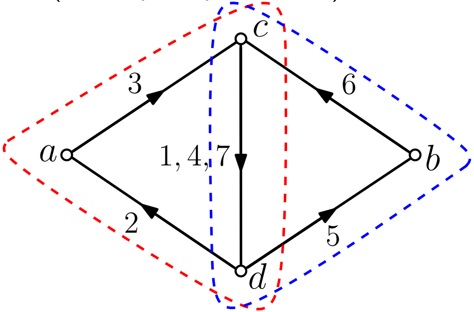
- two different SCCs can have common vertices
  - $\{a, c, d\}$ is a SCC
  - $\{b, c, d\}$ is another SCC
  - $\{a, b, c, d\}$ is not a SCC (no temporal path $b \rightsquigarrow a$)



Temporal strongly connected components

Further differences to the static case:

- two different SCCs can have common vertices
  - $\{a, c, d\}$ is a SCC
  - $\{b, c, d\}$ is another SCC
  - $\{a, b, c, d\}$ is not a SCC (no temporal path $b \rightsquigarrow a$)



- Can we compute/verify temporal SCCs/o-SCCs efficiently?

Temporal strongly connected components

Theorem (Bhadra, Ferreira, 2012)

Given a vertex subset S of a temporal graph (G, λ), we can verify in polynomial time whether S is a SCC (resp. an o-SCC).

Temporal strongly connected components

Theorem (Bhadra, Ferreira, 2012)

Given a vertex subset S of a temporal graph (G, λ), we can verify in polynomial time whether S is a SCC (resp. an o-SCC).

Proof.

- consider the induced (temporal) subgraph on S (resp. whole (G, λ))
- from every vertex $v \in S$ compute all foremost temporal paths
  - or all shortest / fastest paths, with any of the known algorithms
- if at least one vertex v does not reach the whole S:
  - then S is not a SCC (resp. an o-SCC)

□

Observation: similarly to static graphs

Temporal strongly connected components

Theorem

Given a temporal graph (G, λ), it is NP-hard to compute the maximum size of a SCC, even if all edges have one and the same label.

Theorem (Bhadra, Ferreira, 2012)

Given a temporal graph (G, λ), it is NP-hard to compute the maximum size of an o-SCC, even if all edges have two labels.

- Reduction from CLIQUE:
  Input: Graph G
  Goal: Find a clique of maximum size in G.

Overview

- Temporal graphs
- Temporal paths
- Strongly connected components
- Temporal exploration
- Temporal design problems
- Stochastic temporal graphs
- Future research directions

Temporal exploration

Temporal Exploration Problem (TEXP) (Michail, Spirakis, 2014)

Input: *Temporal graph* (G, λ) *and source vertex s*

Goal: *Visit each vertex at least once with a temporal walk that minimizes the arrival time (possibly revisiting vertices)*

Its "static analogue": Graphic Traveling Salesman Problem

- $\frac{13}{9}$-approximation algorithm [Mucha, *Th. Comp. Syst.*, 2014]

Temporal exploration

Temporal Exploration Problem (TEXP) (Michail, Spirakis, 2014)

Input: Temporal graph (G, λ) and source vertex s

Goal: Visit each vertex at least once with a temporal walk that minimizes the arrival time (possibly revisiting vertices)

Its "static analogue": Graphic Traveling Salesman Problem

- $\frac{13}{9}$-approximation algorithm [Mucha, *Th. Comp. Syst.*, 2014]

Observation

The decision version in the static case can be solved in linear time.

Proof.

- A static graph G is explorable $\Leftrightarrow$ G is connected.

$\Rightarrow$ Check connectivity in G by DFS. $\square$

Temporal exploration

Temporal Exploration Problem (TEXP) (Michail, Spirakis, 2014)

Input: *Temporal graph* (G, λ) *and source vertex* s

Goal: *Visit each vertex at least once with a temporal walk that minimizes the arrival time (possibly revisiting vertices)*

Its "static analogue": Graphic Traveling Salesman Problem

- $\frac{13}{9}$-approximation algorithm [Mucha, *Th. Comp. Syst.*, 2014]

Observation

If a temporal graph (G, λ) *is connected at every time* t*, then it is always explorable.*

Temporal exploration

Temporal Exploration Problem (TEXP) (Michail, Spirakis, 2014)

Input: Temporal graph (G, λ) and source vertex s

Goal: Visit each vertex at least once with a temporal walk that minimizes the arrival time (possibly revisiting vertices)

Its "static analogue": Graphic Traveling Salesman Problem

- $\frac{13}{9}$-approximation algorithm [Mucha, *Th. Comp. Syst.*, 2014]

Observation

If a temporal graph (G, λ) is connected at every time t, then it is always explorable.

However:

Theorem (Michail, Spirakis, *MFCS*, 2014)

The decision version in the temporal case is NP-complete.

Temporal exploration

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

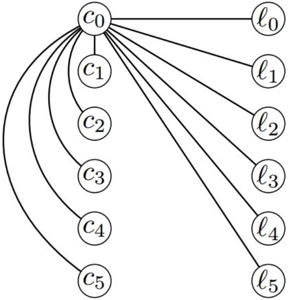
There exists an infinite family of temporal graphs that require $\Omega(n^2)$ time to be explored.

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

There exists an infinite family of temporal graphs that require $\Omega(n^2)$ time to be explored.

Proof.

- Let $V = \{c_j, \ell_j : 0 \leq j \leq n-1\}$ be the vertex set of G
- The "snapshot" of G at time $t \geq 0$ is a star with center $c_{t \bmod n}$



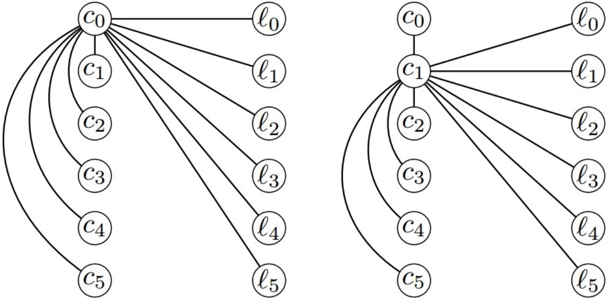
Temporal exploration

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

There exists an infinite family of temporal graphs that require $\Omega(n^2)$ time to be explored.

Proof.

- Let $V = \{c_j, \ell_j : 0 \leq j \leq n-1\}$ be the vertex set of G
- The "snapshot" of G at time $t \geq 0$ is a star with center $c_{t \bmod n}$

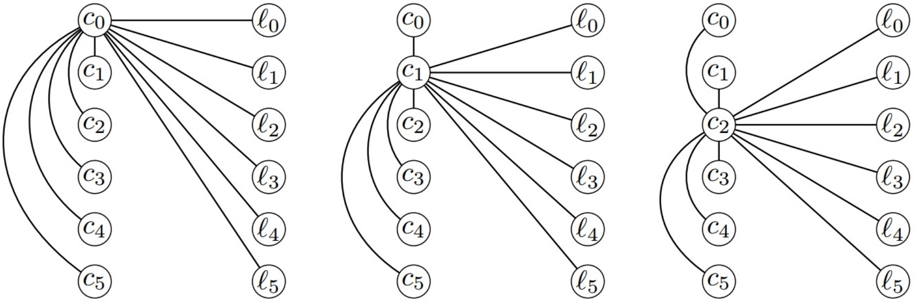


Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

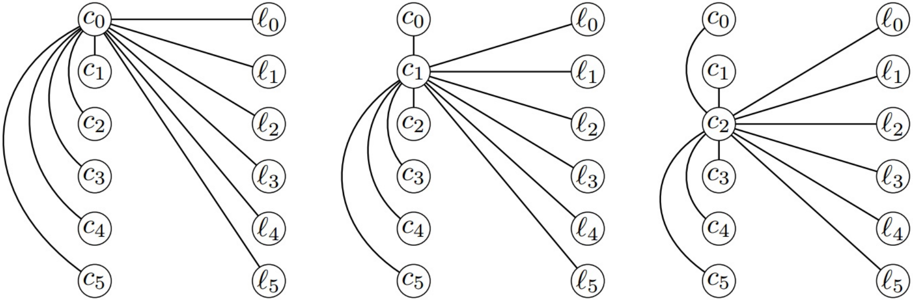
There exists an infinite family of temporal graphs that require $\Omega(n^2)$ time to be explored.

Proof.

- Let $V = \{c_j, \ell_j : 0 \leq j \leq n-1\}$ be the vertex set of G
- The "snapshot" of G at time $t \geq 0$ is a star with center $c_{t \bmod n}$



Proof (continued).

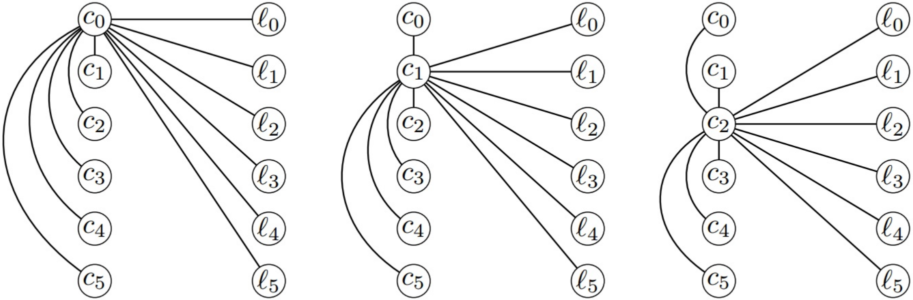


- If the exploring agent is at a vertex that is not the current center:
  - it can only wait or travel to the current center
- If it moves, at the next step it will be again not in the current center

$\Rightarrow$ to go from ℓ_i to ℓ_j, $i \neq j$, n steps are needed:

  - the fastest way is to move from ℓ_i to the current center, to wait $n-1$ steps, and then go to ℓ_j

Proof (continued).



- If the exploring agent is at a vertex that is not the current center:
  - it can only wait or travel to the current center
- If it moves, at the next step it will be again not in the current center

$\Rightarrow$ to go from ℓ_i to ℓ_j, $i \neq j$, n steps are needed:
  - the fastest way is to move from ℓ_i to the current center, to wait $n-1$ steps, and then go to ℓ_j

$\Rightarrow$ the total number of steps is $\Omega(n^2)$

Temporal exploration

Modifying the reduction used to prove NP-completeness, the result can be strongly amplified:

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

Approximating TEXP with ratio $O(n^{1-\varepsilon})$ is NP-hard.

Temporal exploration

Modifying the reduction used to prove NP-completeness, the result can be strongly amplified:

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

Approximating TEXP with ratio $O(n^{1-\varepsilon})$ is NP-hard.

The previous reduction can be also "parameterized":

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

For every $\Delta > 0$, there exists an infinite family of temporal graphs with maximum degree Δ that require $\Omega(\Delta n)$ time to be explored.

Temporal exploration

Modifying the reduction used to prove NP-completeness, the result can be strongly amplified:

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

Approximating TEXP with ratio $O(n^{1-\varepsilon})$ is NP-hard.

The previous reduction can be also "parameterized":

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

For every $\Delta > 0$, there exists an infinite family of temporal graphs with maximum degree Δ that require $\Omega(\Delta n)$ time to be explored.

Furthermore, on restricted classes of underlying graphs:

Theorem (Erlebach, Hoffmann, Kammer, *ICALP*, 2015)

Any temporal graph whose underlying graph has treewidth at most k, can be explored in $O(n^{1.5}k^2 \log n)$ time.

- Restricting the problem on the special class of underlying star graphs.
- Motivation: inspired by the well-known Traveling Salesperson Problem (TSP): "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin?", i.e. find a min-cost Hamiltonial cycle.
  - What if the salesperson has particular temporal constraints, e.g. (s)he can only go from city A to city B on Mondays or Tuesdays?
  - What if (s)he needs to return to their home town after visiting each city?
  - Can the salesperson decide whether (s)he can visit all towns and finally return to their home town by a certain day or time?

Definition (Temporal Star)

A temporal star is a temporal graph (G_s, λ) on a star graph $G_s = (V, E)$.

Definition (Exploration of a temporal star)

A (partial) exploration of a temporal star is a journey J that starts and ends at the center of G_s which visits some nodes of G_s; its size $|J|$ is the number of nodes of G_s that are visited by J.

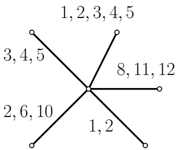
Definition (Temporal Star)

A temporal star is a temporal graph (G_s, λ) on a star graph $G_s = (V, E)$.

Definition (Exploration of a temporal star)

A (partial) exploration of a temporal star is a journey J that starts and ends at the center of G_s which visits some nodes of G_s; its size $|J|$ is the number of nodes of G_s that are visited by J.

- We "enter" (resp. "exit") an edge when we cross it from center to leaf (resp. leaf to center) at a time on which the edge is available.
- We can assume that in an exploration the entry to any edge e is followed by the exit from e at the earliest possible time. Waiting at a leaf (instead of exiting as soon as possible) does not help in exploring more edges.



Star Exploration Problem ($\textsc{StarExp}(k)$) (Akrida, Mertzios, Spirakis, *CIAC*, 2019)

Input: *A temporal star* (G_s, λ) *such that every edge has at most k labels.*

Question: *Is* (G_s, λ) *explorable?*

Maximum Star Exploration Problem ($\textsc{MaxStarExp}(k)$) (Akrida et al., *CIAC*, 2019)

Input: *A temporal star* (G_s, λ) *such that every edge has at most k labels.*

Output: *A (partial) exploration of* (G_s, λ) *of maximum size.*

- $\textsc{MaxStarExp}(3)$ can be efficiently solved in $O(n \log n)$ time
- $\textsc{StarExp}(k)$ is NP-complete, when $k \geq 6$
- $\textsc{MaxStarExp}(k)$ is APX-complete, when $k \geq 4$

	Maximum number of labels per edge					
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k \geq 6$
$\textsc{StarExp}(k)$	No	$O(n \log n)$	$O(n \log n)$	?	?	NP-c
$\textsc{MaxStarExp}(k)$	No	$O(n \log n)$	$O(n \log n)$	APX-c	APX-c	APX-c

Theorem (Akrida, Mertzios, Spirakis, *CIAC*, 2019)

$\textsc{MaxStarExp}(3)$ *can be efficiently solved in* $O(n \log n)$ *time*

Proof (sketch).

- problem is reducible to the Interval Scheduling Maximization Problem (ISMP):
  Input: A set of intervals, each with a start and a finish time.
  Output: Find a max-size set of non-overlapping intervals.

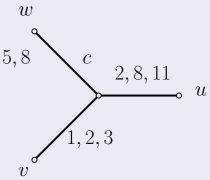
Theorem (Akrida, Mertzios, Spirakis, *CIAC*, 2019)

$\textsc{MaxStarExp}(3)$ *can be efficiently solved in* $O(n \log n)$ *time*

Proof (sketch).

- problem is reducible to the Interval Scheduling Maximization Problem (ISMP):
  Input: A set of intervals, each with a start and a finish time.
  Output: Find a max-size set of non-overlapping intervals.

- Every edge e can be viewed as one or two intervals to be scheduled.
  - Any edge with a single label can be ignored.



Interval 1: [2,8]
Interval 2: [8,11]
Interval 3: [1,2]
Interval 4: [2,3]
Interval 5: [5,8]

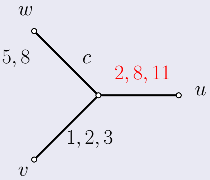
Theorem (Akrida, Mertzios, Spirakis, *CIAC*, 2019)

$\textsc{MaxStarExp}(3)$ *can be efficiently solved in* $O(n \log n)$ *time*

Proof (sketch).

- problem is reducible to the Interval Scheduling Maximization Problem (ISMP):
  Input: A set of intervals, each with a start and a finish time.
  Output: Find a max-size set of non-overlapping intervals.

- Every edge e can be viewed as one or two intervals to be scheduled.
  - Any edge with a single label can be ignored.



Interval 1: [2,8]
Interval 2: [8,11]
Interval 3: [1,2]
Interval 4: [2,3]
Interval 5: [5,8]

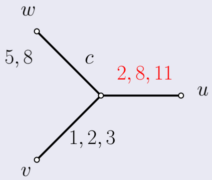
Theorem (Akrida, Mertzios, Spirakis, *CIAC*, 2019)

$\textsc{MaxStarExp}(3)$ *can be efficiently solved in* $O(n \log n)$ *time*

Proof (sketch).

- problem is reducible to the Interval Scheduling Maximization Problem (ISMP):
  Input: A set of intervals, each with a start and a finish time.
  Output: Find a max-size set of non-overlapping intervals.

- Any (partial) exploration of (G_s, λ) corresponds to a set of non-overlapping intervals of the same size as the exploration, and vice versa.



Interval 1: [2,8]
Interval 2: [8,11]
Interval 3: [1,2]
Interval 4: [2,3]
Interval 5: [5,8]

Temporal exploration

The travelling salesperson who returns home

Theorem (Akrida, Mertzios, Spirakis, *CIAC*, 2019)

$\textsc{MaxStarExp}(3)$ *can be efficiently solved in* $O(n \log n)$ *time*

Proof (sketch, continued).

Greedy optimal solution for ISMP:

1. Start with the set $\mathcal{I}$ of all intervals $(|\mathcal{I}| \leq 2(n-1))$. Select the iterval, I, with the earliest finish time.
2. Remove from $\mathcal{I}$ the interval I and all overlapping intervals.
3. Repeat until $\mathcal{I}$ is empty.

Time needed: $(|\mathcal{I}| \log |\mathcal{I}|) = O(n \log n)$ □

Temporal exploration: further work

- Minimum cost exploration; complete directed temporal graph with edge weights from $\{1, 2\}$
  [Michail and Spirakis. Traveling salesman problems in temporal graphs, *TCS*, 2016.]
- Exploration of constantly connected dynamic graphs; underlying cactus graph
  [Ilcinkas,Klasing, Wade. Exploration of constantly connected dynamic graphs based on cactuses, *SIROCCO*, 2014.]
- Exploration of temporal graphs of small pathwidth; NP-completeness
  [Bodlaender and van der Zanden. On exploring always-connected temporal graphs of small pathwidth, *Information Processing Letters*, 2014.]
- Exploration of temporal graphs using temporal paths with non-strictly increasing labels
  [Erlebach, Spooner. Non-strict Temporal Exploration, *SIROCCO*, 2020.]

Overview

- Temporal graphs
- Temporal paths
- Strongly connected components
- Temporal exploration
- Temporal design problems
- Stochastic temporal graphs
- Future research directions

Temporal design problems

So far:

- we were given the input temporal graph (G, λ) and
- we were asked to optimize some metric (e.g. a foremost path)

Many times the problem is different:

- we are given a graph G and
- we are asked to construct a time-labeling λ such that:
  - λ minimizes some cost function and
  - (G, λ) satisfies some connectivity constraints

[Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017]

[Mertzios, Michail, Spirakis, *Algorithmica*, 2019]

Temporal design problems

In many scheduling problems:

- the provided graph topology G represents a given static specification
  - e.g. available bus routes in the city center
- the aim is to organize a temporal schedule on this specification, e.g.
  - when the buses should be in which stop
  - such that every pair of stops is connected via a route
- while minimizing some cost function
  - e.g. with as few buses as possible

Temporal design problems

In many scheduling problems:

- the provided graph topology G represents a given static specification
  - e.g. available bus routes in the city center
- the aim is to organize a temporal schedule on this specification, e.g.
  - when the buses should be in which stop
  - such that every pair of stops is connected via a route
- while minimizing some cost function
  - e.g. with as few buses as possible

Creating and maintaining a connection does not come for free, e.g.:

- edge "rentals" / toll roads
- in wireless sensor networks the connection cost depends on the power consumption of the vertices awake

Temporal design problems

We mainly study the following cost functions of a time-labeling λ:

1. temporal cost κ: the total number of labels on all edges
   - a centralized measure of cost
2. temporality τ: the maximum number of labels per edge
   - a distributed / decentralized measure of cost in the temporal network
3. as well as trade-offs between the age $\alpha(\lambda)$ and these parameters

Temporal design problems

We mainly study the following cost functions of a time-labeling λ:

1. temporal cost κ: the total number of labels on all edges
   - a centralized measure of cost
2. temporality τ: the maximum number of labels per edge
   - a distributed / decentralized measure of cost in the temporal network
3. as well as trade-offs between the age $\alpha(\lambda)$ and these parameters

and two fundamental connectivity properties:

1. preserve in (G, λ) all reachabilities in G
   - if v is reachable from u in $G \Rightarrow u \rightsquigarrow v$ in (G, λ)
2. preserve in (G, λ) all paths in G
   - G has a path $P \Rightarrow (G, \lambda)$ has a temporal path on the same edges as P

Temporal design problems

We mainly study the following cost functions of a time-labeling λ:

1. temporal cost κ: the total number of labels on all edges
   - a centralized measure of cost
2. temporality τ: the maximum number of labels per edge
   - a distributed / decentralized measure of cost in the temporal network
3. as well as trade-offs between the age $\alpha(\lambda)$ and these parameters

and two fundamental connectivity properties:

1. preserve in (G, λ) all reachabilities in G
   - if v is reachable from u in $G \Rightarrow u \rightsquigarrow v$ in (G, λ)
2. preserve in (G, λ) all paths in G
   - G has a path $P \Rightarrow (G, \lambda)$ has a temporal path on the same edges as P

Notation (combining cost function & connectivity property):

- $\kappa(G, \mathit{reach})$, $\tau(G, \mathit{all\ paths})$, $\tau(G, \mathit{all\ paths}, \alpha(\lambda))$, etc.

Temporal cost for preserving all reachabilities

Cost function: total number κ of labels

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

Let $d(G)$ denote the (static) diameter of the directed graph G. The problem of computing $\kappa(G, reach, d(G))$ is APX-hard, even when each directed cycle of G has length at most 2.

Temporal cost for preserving all reachabilities

Cost function: total number κ of labels

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

Let $d(G)$ denote the (static) diameter of the directed graph G. The problem of computing $\kappa(G, reach, d(G))$ is APX-hard, even when each directed cycle of G has length at most 2.

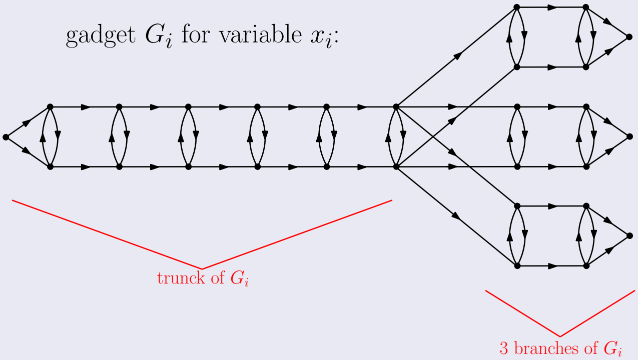
Proof (sketch).

- reduction from Max-XOR(3):
  - formula ϕ with n variables and m clauses
  - XOR-clauses $(\ell_i \oplus \ell_j)$ with two literals each: $(\ell_i \oplus \ell_j) = 1 \;\Leftrightarrow\; \ell_i \neq \ell_j$
  - each variable appears in at most 3 clauses $\;\Rightarrow\;$ $m \leq \frac{3}{2}n$
  - the goal is to find a truth assignment τ with the maximum number $|\tau(\phi)|$ of XOR-satisfied clauses
- from ϕ we construct a graph G_ϕ and we prove:
  - $|\tau(\phi)| \geq k \;\Leftrightarrow\; \kappa(G_\phi, reach, d(G_\phi)) \leq 39n - 4m - 2k$

Temporal cost for preserving all reachabilities

Proof (sketch, continued).

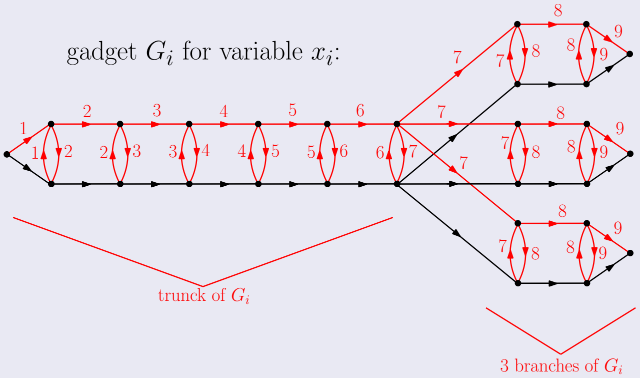
- diameter $d(G_i) = 9$



Temporal cost for preserving all reachabilities

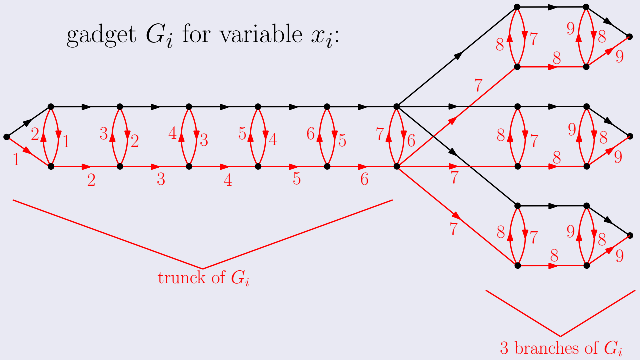
Proof (sketch, continued).

- diameter $d(G_i) = 9$
- to achieve a maximum label 9 we have two choices:
  - $x_i = 0$



Proof (sketch, continued).

- diameter $d(G_i) = 9$
- to achieve a maximum label 9 we have two choices:
  - $x_i = 0$
  - $x_i = 1$

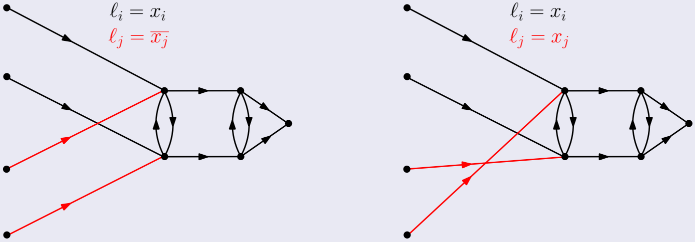


Temporal cost for preserving all reachabilities

Proof (sketch, continued).

- for every clause $(\ell_i \oplus \ell_j)$ where:
  - ℓ_i corresponds to the pth appearance of x_i $\quad (p \in \{1,2,3\})$
  - ℓ_j corresponds to the qth appearance of x_j $\quad (q \in \{1,2,3\})$

  we identify the pth branch of G_i and the qth branch of G_j as follows:

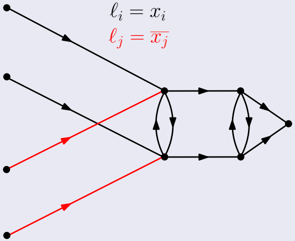


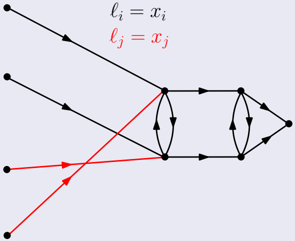
□

Proof (sketch, continued).

- for every clause $(\ell_i \oplus \ell_j)$ where:
  - ℓ_i corresponds to the pth appearance of x_i $\quad (p \in \{1, 2, 3\})$
  - ℓ_j corresponds to the qth appearance of x_j $\quad (q \in \{1, 2, 3\})$

  we identify the pth branch of G_i and the qth branch of G_j as follows:





- $\ell_i \neq \ell_j \;\Leftrightarrow\;$ the correct "tracks" of these branches are labeled
- otherwise we use both "tracks" $\;\Rightarrow\;$ pay more labels

$\square$

Temporal cost for preserving all reachabilities

A simple approximation algorithm:

- the reachability number of $u \in V$:

$$r(u) = |\{v \in V : v \text{ is reachable from } u\}|$$

- the total reachability number: $r(G) = \sum_{u \in V} r(u)$

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

A $\frac{r(G)}{n-1}$-approximation for $\kappa(G, reach, d(G))$ can be computed in polynomial time for connected graphs G.

Temporal cost for preserving all reachabilities

A simple approximation algorithm:

- the reachability number of $u \in V$:

$$r(u) = |\{v \in V : v \text{ is reachable from } u\}|$$

- the total reachability number: $r(G) = \sum_{u \in V} r(u)$

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

A $\frac{r(G)}{n-1}$-approximation for $\kappa(G, reach, d(G))$ can be computed in polynomial time for connected graphs G.

Proof.

- Compute from every $u \in V$ a temporal out-tree

$\Rightarrow$ all reachabilities are maintained with $\leq r(G)$ labels

- OPT$\geq n - 1 \;\Rightarrow$ approximation ratio $\frac{r(G)}{n-1}$ $\square$

Nearly cost-optimal design for preserving all reachabilities

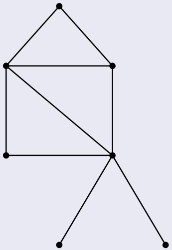
Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

Given a connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices, we can construct a labelling λ of cost $c(\lambda) = 2n - 3$ that preserves all reachabilities on G in polynomial time.

Proof.

- We consider a fixed, arbitrary spanning tree T of G and let a leaf node w be its root.

$G = (V, E)$



$\square$

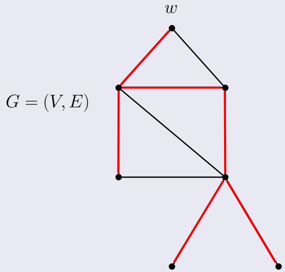
Nearly cost-optimal design for preserving all reachabilities

Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

Given a connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices, we can construct a labelling λ of cost $c(\lambda) = 2n - 3$ that preserves all reachabilities on G in polynomial time.

Proof.

- We consider a fixed, arbitrary spanning tree T of G and let a leaf node w be its root.



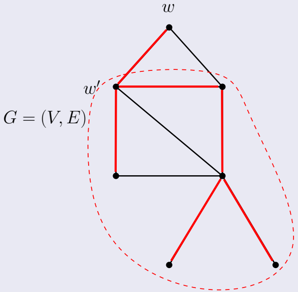
Nearly cost-optimal design for preserving all reachabilities

Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

Given a connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices, we can construct a labelling λ of cost $c(\lambda) = 2n - 3$ that preserves all reachabilities on G in polynomial time.

Proof.

- We consider a fixed, arbitrary spanning tree T of G and let a leaf node w be its root.
- Let w' be the single child of w in T and let T' be the subtree of T that is rooted at w'.



□

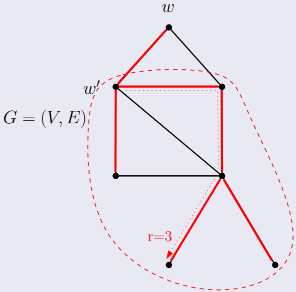
Nearly cost-optimal design for preserving all reachabilities

Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

Given a connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices, we can construct a labelling λ of cost $c(\lambda) = 2n - 3$ that preserves all reachabilities on G in polynomial time.

Proof.

- We consider a fixed, arbitrary spanning tree T of G and let a leaf node w be its root.
- Let w' be the single child of w in T and let T' be the subtree of T that is rooted at w'.
- Let r length of longest path from w' to any leaf of T'.



□

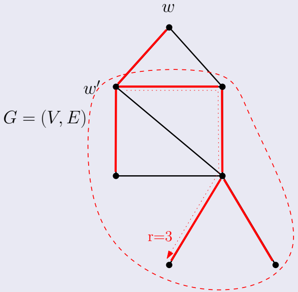
Nearly cost-optimal design for preserving all reachabilities

Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

Given a connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices, we can construct a labelling λ of cost $c(\lambda) = 2n - 3$ that preserves all reachabilities on G in polynomial time.

Proof.

- We consider a fixed, arbitrary spanning tree T of G and let a leaf node w be its root.
- Let w' be the single child of w in T and let T' be the subtree of T that is rooted at w'.
- Let r length of longest path from w' to any leaf of T'.
- We assign labels to the edges of T as follows.



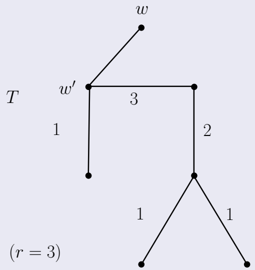
□

Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

Given a connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices, we can construct a labelling λ of cost $c(\lambda) = 2n - 3$ that preserves all reachabilities on G in polynomial time.

Proof.

Going upwards.
Any edge of T' incident to a leaf gets label 1. Any edge $e = \{u, v\}$ of T', with $d(w', v) = d(w', u) + 1$, where the subtree T^* rooted at v has been labelled going upwards towards w', gets a label $l_e = max\{\text{all labels in } T^*\} + 1$.



$\square$

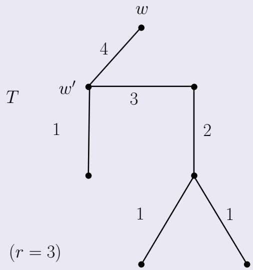
Nearly cost-optimal design for preserving all reachabilities

Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

Given a connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices, we can construct a labelling λ of cost $c(\lambda) = 2n - 3$ that preserves all reachabilities on G in polynomial time.

Proof.

The edge $\{w, w'\}$.
We label the edge $\{w, w'\}$ of T with the single label $r + 1$.



$\square$

Nearly cost-optimal design for preserving all reachabilities

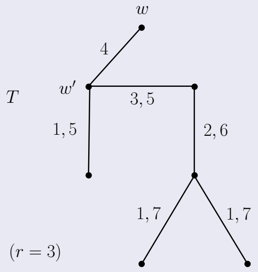
Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

Given a connected undirected graph $G = (V, E)$ of $n \geq 2$ vertices, we can construct a labelling λ of cost $c(\lambda) = 2n - 3$ that preserves all reachabilities on G in polynomial time.

Proof.

Going downwards.

Any edge of T' incident to w' gets a label $r + 2$. Any edge e of T' in a path from w' to a leaf of T', the *parent edge* of which has been labelled, going downwards, with label l', gets a label $l_e = l' + 1$.



$\square$

Removal cost for preserving all reachabilities

The "inverse" design problem:

- given a temporal graph (G, λ) that maintains all reachabilities of G
- remove the maximum number of labels by maintaining reachabilities
- removal cost $r(G, \lambda)$

Removal cost for preserving all reachabilities

Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

The problem of computing $r(G, \lambda)$ is APX-hard on undirected graphs G.

Removal cost for preserving all reachabilities

Theorem (Akrida, Gąsieniec, Mertzios, Spirakis, *TOCS*, 2017)

The problem of computing $r(G, \lambda)$ is APX-hard on undirected graphs G.

Proof (sketch).

- reduction from monotone Max-XOR(3):
  - same as Max-XOR(3) but no variable is negated
- from ϕ we construct a graph G_ϕ and we prove:
  - $|\tau(\phi)| \geq k \;\Leftrightarrow\; r(G_\phi, \lambda) \geq 9n + k$
  - assuming a PTAS for computing $r(G_\phi, \lambda)$, we obtain a PTAS for monotone Max-XOR(3)
  - Contradiction; monotone Max-XOR(3) is APX-hard [Alimonti and Kann, *CIAC*, 1997]

Proof (sketch, continued).

Construction of G_ϕ, λ

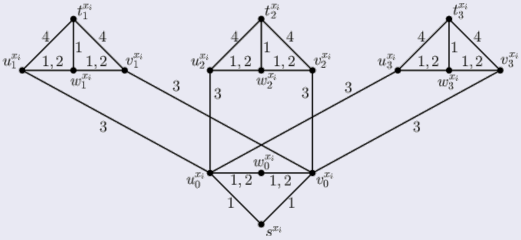


Figure: The gadget $G_{\phi,i}$ for the variable x_i.

- For every $p \in \{1, 2, 3\}$ we associate the pth appearance of the variable x_i in a clause of ϕ with the pth branch of $G_{\phi,i}$.

Removal cost for preserving all reachabilities

Proof (sketch, continued).

Construction of G_ϕ, λ

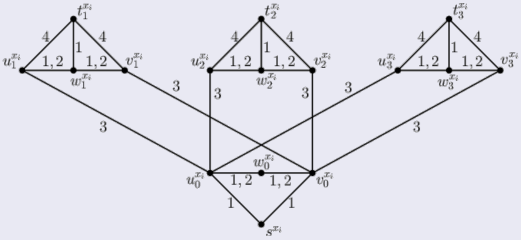


Figure: The gadget $G_{\phi,i}$ for the variable x_i.

- For every pair of vertices $w_p^{x_i}, w_q^{x_j}$, $p, q \in \{0, 1, 2, 3\}$, $i, j \in \{1, 2, \ldots, n\}$ add an edge $e = \{w_p^{x_i}, w_q^{x_j}\}$ with label 7.

Removal cost for preserving all reachabilities

Proof (sketch, continued).

Construction of G_ϕ, λ

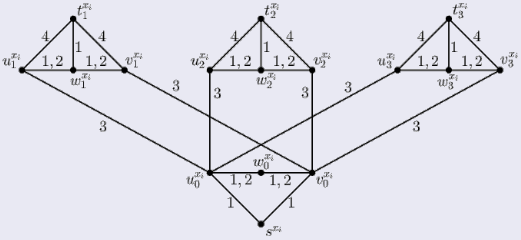


Figure: The gadget $G_{\phi,i}$ for the variable x_i.

- For every pair of vertices $t_p^{x_i}, t_q^{x_j}$, $p, q \in \{1, 2, 3\}$, $i, j \in \{1, 2, \ldots, n\}$ add an edge with label 7.

Removal cost for preserving all reachabilities

Proof (sketch, continued).

Construction of G_ϕ, λ

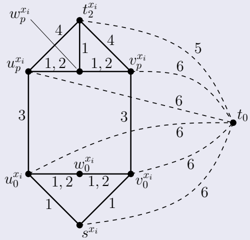


Figure: The addition of vertex t_0. There exists in G_ϕ also the edge $\{t_0, w_0^{x_n}\}$ with label 5.

- Add vertex t_0 adjacent to all vertices $\{s^{x_i}, t_1^{x_i}, t_2^{x_i}, t_3^{x_i}, u_p^{x_i}, v_p^{x_i} : 1 \leq i \leq n,\ 0 \leq p \leq 3\}$ with label 5 or 6.

Removal cost for preserving all reachabilities

Proof (sketch, continued).

Construction of G_ϕ, λ

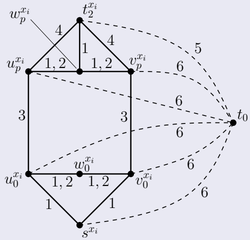


Figure: The addition of vertex t_0. There exists in G_ϕ also the edge $\{t_0, w_0^{x_n}\}$ with label 5.

- Add the edge $t_0, w_0^{x_n}$ with label 5.

Removal cost for preserving all reachabilities

Proof (sketch, continued).

Construction of G_ϕ, λ

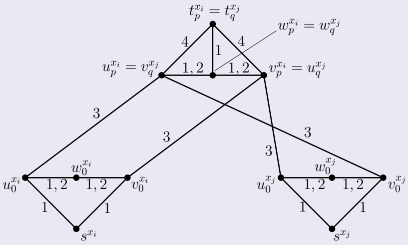


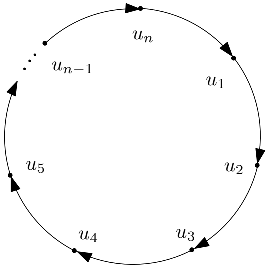
Figure: The gadget for the clause $(x_i \oplus x_j)$.

- Consider now a clause $\alpha = (x_i \oplus x_j)$ of ϕ. Assume that the variable x_i (resp. x_j) of α corresponds to the pth (resp. to the qth) appearance of x_i (resp. of x_j) in ϕ. Then we identify the vertices $u_p^{x_i}, v_p^{x_i}, w_p^{x_i}, t_p^{x_i}$ of the pth branch of $G_{\phi,i}$ with the vertices $v_q^{x_i}, u_q^{x_i}, w_q^{x_i}, t_q^{x_i}$ of the qth branch of $G_{\phi,j}$, respectively.

□

Temporality of the ring C_n

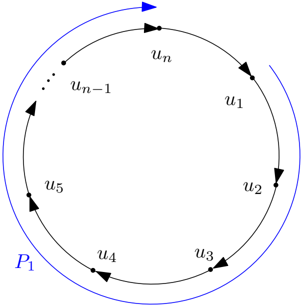
A very different cost function: maximum number τ of labels per edge



Temporality of the ring C_n

A very different cost function: maximum number τ of labels per edge

- increasing labels on P_1 $\Rightarrow$ decreasing labels from (u_{n-1}, u_n) to (u_1, u_2)



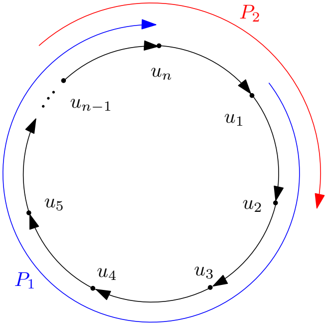
Temporality of the ring C_n

A very different cost function: maximum number τ of labels per edge

- increasing labels on P_1 $\Rightarrow$ decreasing labels from (u_{n-1}, u_n) to (u_1, u_2)
- P_2 uses first (u_{n-1}, u_n), then (u_1, u_2)

$\Rightarrow$ increasing pair of labels on these edges

- To preserve both P_1, P_2 we need 2 labels on at least one of these two edges $\Rightarrow$ $\tau(C_n, \textit{all paths}) \geq 2$



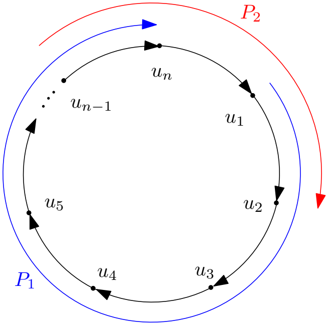
Temporality of the ring C_n

A very different cost function: maximum number τ of labels per edge

- increasing labels on P_1 $\Rightarrow$ decreasing labels from (u_{n-1}, u_n) to (u_1, u_2)
- P_2 uses first (u_{n-1}, u_n), then (u_1, u_2)

$\Rightarrow$ increasing pair of labels on these edges

- To preserve both P_1, P_2 we need 2 labels on at least one of these two edges $\Rightarrow$ $\tau(C_n, \textit{all paths}) \geq 2$



- The labeling that assigns to each edge (u_i, u_{i+1}) the labels $\{i, n+i\}$ preserves all simple paths, i.e. $\tau(C_n, \textit{all paths}) \leq 2$

$\Rightarrow$ $\tau(C_n, \textit{all paths}) = 2$

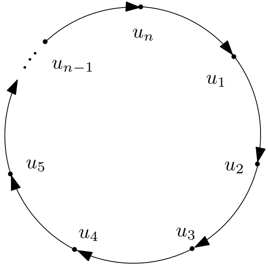
- The maximum label is $2n$ (can be "tuned" to $2n-2$)

Temporality of the ring C_n

Restricting the age

What if we restrict the age to $\alpha(\lambda) = n - 1$?

- Assume that some edge e of C_n misses label $i \in \{1, 2, \ldots, n-1\}$
- Then there exists a temporal path on C_n that needs label i on edge e to finish by time $n - 1$



Temporality of the ring C_n

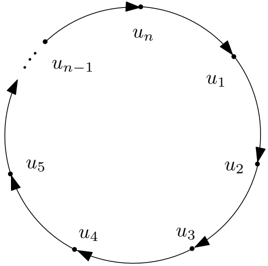
Restricting the age

What if we restrict the age to $\alpha(\lambda) = n-1$?

- Assume that some edge e of C_n misses label $i \in \{1, 2, \ldots, n-1\}$
- Then there exists a temporal path on C_n that needs label i on edge e to finish by time $n-1$

$\Rightarrow$ the optimal labeling assigns $\{1, 2, , \ldots, n-1\}$ to all edges of C_n

$\Rightarrow$ $\tau(C_n, \textit{all paths}, n-1) = n-1$



More generally:

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

If G is a directed ring C_n and $\alpha(\lambda) = (n-1) + k$, where $1 \leq k \leq n-1$, then

$$\tau(G, \textit{all paths}, \alpha) = \Theta(n/k).$$

In particular: $\lfloor \frac{n-1}{k+1} \rfloor + 1 \leq \tau(G, \textit{all paths}, \alpha) \leq \lceil \frac{n}{k+1} \rceil + 1.$

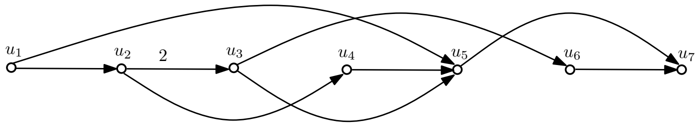
Temporality of a DAG

A topological sort of a digraph G:

- a linear ordering of its vertices, where
- if G contains an arc (u, v) then u appears before v

It is known:

- a digraph G can be topologically sorted $\Leftrightarrow$ G is a DAG



Temporality of a DAG

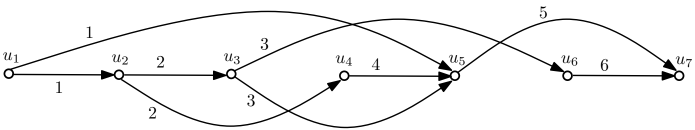
Lemma (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

If G is a DAG then $\tau(G, \text{all paths}) = 1$.

Proof.

- Take a topological sort $u_1, u_2, \ldots, u_n$ of G
- Give label i to every edge (u_i, u_j), where $i < j$.

$\square$



Temporality: preserving all reachabilities

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

Let G be an undirected (or strongly connected directed) graph. Then $\tau(G, reach) \leq 2$.

Proof.

- pick an arbitrary vertex v
- let v have (static) distance at most k to all other vertices
- build a temporal in-tree to vertex v with labels $\{1, 2, \dots, k\}$
- from v build a temporal out-tree to vertex v with labels $\{k+1, k+2, \dots, 2k\}$

$\Rightarrow$ all vertices remain temporally connected with 2 labels per edge

$\square$

Temporality: preserving all reachabilities

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

Let G be an undirected (or strongly connected directed) graph. Then $\tau(G, reach) \leq 2$.

Similarly to the analysis for DAGs:

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

Let G be a directed graph. Then $\tau(G, reach) = \max_{C \in \mathcal{C}(G)} \tau(C, reach)$, where $\mathcal{C}(G)$ is the set of strongly connected components of G.

Temporality: preserving all reachabilities

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

Let G be an undirected (or strongly connected directed) graph. Then $\tau(G, reach) \leq 2$.

Similarly to the analysis for DAGs:

Theorem (Mertzios, Michail, Spirakis, *Algorithmica*, 2019)

Let G be a directed graph. Then $\tau(G, reach) = \max_{C \in \mathcal{C}(G)} \tau(C, reach)$, where $\mathcal{C}(G)$ is the set of strongly connected components of G.

Therefore:

Corollary

$\tau(G, reach) \leq 2$ for every directed or undirected graph.

That is: we can preserve all reachabilities with at most 2 labels per edge.

Overview

- Temporal graphs
- Temporal paths
- Strongly connected components
- Temporal design problems
- Temporal exploration
- Stochastic temporal graphs
- Future research directions

Stochastic temporal graphs

Levels of knowledge about the network evolution:

- whole temporal graph given in advance
- adversary who reveals it snapshot-by-snapshot at every time step
- intermediate knowledge setting, captured by **stochastic temporal graphs**, where the network evolution is given by a probability distribution that governs the appearance of each edge over time

Stochastic temporal graphs

Models of randomness:

- "Selection from a pool of labels": the labels assigned to the edges of the underlying graph are chosen independently and uniformly at random from a set of available time labels.
  - ephemeral random dynamic networks
- "Fixed probability": each edge exists at each time-step with a certain probability
  - opportunistic mobile networks
- "Memory effect": appearance probability of a particular edge at a given time-step t depends on the appearance (or absence) of the same edge at the previous $k \geq 1$ time steps
  - faulty network communication

- Upper and lower bounds on the Temporal Diameter, i.e., expected maximum temporal distance, of the complete graph with a single label per edge
  - Note that the complete graph is the only graph for which *preserving all reachabilities* is guaranteed with random labels, even with 1 label per edge.

  [Akrida, Gąsieniec, Mertzios, Spirakis *JPDC*, 2016]
- Bound on the smallest number of random labels per edge that guarantees preservation of reachability with high probability ⇒ upper bound on a measure of how "good" the best random assignment is compared to the best deterministic one.
  [Akrida, Gąsieniec, Mertzios, Spirakis *JPDC*, 2016]
- High removal profit with high probability for complete graphs and random graphs.
  [Akrida, Gąsieniec, Mertzios, Spirakis *TOCS*, 2017]
- Flows in uniform random temporal networks: characterisation of networks that block any flow from arriving to the target.
  [Akrida, Czyzowicz, Gąsieniec, Kuszner, Spirakis, *J. Comp. Syst. Sci.*, 2019]

Stochastic temporal graphs
Fixed probability per time step

- Model of opportunistic mobile networks as a type of random temporal networks, where each edge exists at each time-step with a fixed probability; proof of existence of small diameter.
  [Chaintreau, Mtibaa, Massoulié, Diot, *CoNEXT*, 2007]
- Dijkstra-like polynomial-time algorithm for computing the arrival time of a best policy for choosing an source-target journey
  [Basu, Bar-Noy, Ramanathan, Johnson, *arXiv*, 2010]
- Cost-effective data dissemination approach in disruption tolerant networks, based on a centrality metric.
  [Gao, Cao, *INFOCOM*, 2011]
- Efficient content dissemination in opportunistic social networks broken down into 'temporal communities', i.e., groups of people meeting periodically.
  [Pietiläinen, Diot, *MOBIHOC*, 2012]

Stochastic temporal graphs

Probability per time step with dependence of history

- Case of each edge existing at each time-step with probability dependent on the previous time-step: upper bounds for the flooding time and tight characterizations of the graphs on which the flooding time is constant
  [Clementi, Macci, Monti, Pasquale, Silvestri, *SIDMA*, 2010]

- Investigation of the complexity of two temporal path problems, namely computing the expected arrival time of a foremost source-target journey and of a best policy for choosing a particular source-target journey.
  [Akrida, Mertzios, Nikoletseas, Raptopoulos, Spirakis, Zamaraev, *J. Comp. Syst. Sci.*, 2020]

- Temporal graphs
- Temporal paths
- Strongly connected components
- Temporal design problems
- Temporal exploration
- Stochastic temporal graphs
- Future research directions

Research Directions

- Find constant-factor approximations for the various temporal graph design problems
- Other natural connectivity properties subject to which optimization is to be performed
- Efficient deterministic/randomized/approximation algorithms on special temporal graph classes, i.e. by restricting:
  - the underlying topology G and/or
  - the temporal pattern with which the time-labels appear
    (a new dimension with no previous static analogue!)

Research Directions

- Natural non-path temporal problems
  - the notion of a "Δ-temporal clique" in social networks:
    "a set of nodes and a time interval such that all pairs interact at least every Δ during this interval"
    [Viard, Latapy, Magnien, *ASONAM*, 2015]

Research Directions

- Natural non-path temporal problems
  - the notion of "temporal matchings" has been studied before:
    "a collection of edges and time steps of their existence such that the edges are a matching and each edge appears at a different time-step"
    [Michail, Spirakis, *TCS*, 2016]

Research Directions

- Natural non-path temporal problems
  - more recently defined temporal analogues of "vertex cover":
    "a collection of vertex appearances that cover every edge at least once" and "a collection of vertex appearances and a time length so that every edge is covered at least once within every time interval of the given length"
    [Akrida, Mertzios, Spirakis, Zamaraev, *J. Comp. Syst. Sci.*, 2020]

Research Directions

- Natural non-path temporal problems
  - more recently defined temporal analogues of "vertex cover":
    "a collection of vertex appearances that cover every edge at least once" and "a collection of vertex appearances and a time length so that every edge is covered at least once within every time interval of the given length"
    [Akrida, Mertzios, Spirakis, Zamaraev, *J. Comp. Syst. Sci.*, 2020]

- Our results so far are a first step towards answering this fundamental question:

  To what extent can algorithmic and structural results
  of graph theory be carried over to temporal graphs?

Thank you for your attention!