Temporal Graph Mining

Aristides Gionis, Lutz Oettershagen, Ilie Sarpe
{argioni, lutzo, ilsarpe}@kth.se
KTH Royal Institute of Technology
Tutors

Aristides Gionis

Lutz Oettershagen

Ilie Sarpe
Website

https://miningtemporalnetworks.github.io/
menti.com

code: 14 46 97 6
Agenda

Part I: Introduction and Motivation
- models of temporal networks
- algorithmic approaches

Part II: Mining Temporal Networks A:
- connectivity, temporal properties
- centrality, cores

Part III: Mining Temporal Networks B:
- communities, patterns and events
- diffusion and random networks

Part IV: Tools and Code Libraries

Part V: Challenges, Open Problems, and Trends
Part I

Introduction and Motivation
Interconnected World

- networks model objects and their relations
Interconnected World

▶ networks model objects and their relations

▶ many different network types
  - social (WhatsApp, LinkedIn, etc.)
Interconnected World

- networks model objects and their relations

- many different network types
  - social (WhatsApp, LinkedIn, etc..)
  - informational (co-authorship, email, etc..)
Interconnected World

- networks model **objects** and their **relations**

- many different **network types**
  - social (WhatsApp, LinkedIn, etc..)
  - informational (co-authorship, email, etc..)
  - technological (IP-level, transportation, etc..)
Interconnected World

- networks model objects and their relations

- many different network types
  - social (WhatsApp, LinkedIn, etc.)
  - informational (co-authorship, email, etc.)
  - technological (IP-level, transportation, etc.)
  - purchase (E-commerce, crypto, etc.)
Interconnected World

- networks model **objects** and their **relations**

- **many different** **network types**
  - social (WhatsApp, LinkedIn, etc..)
  - informational (co-authorship, email, etc..)
  - technological (IP-level, transportation, etc..)
  - purchase (E-commerce, crypto, etc..)
  - ...
Interconnected World

- networks model objects and their relations

- many different network types
  - social (WhatsApp, LinkedIn, etc.)
  - informational (co-authorship, email, etc.)
  - technological (IP-level, transportation, etc.)
  - purchase (E-commerce, crypto, etc.)
  - ...

- networks are ubiquitous in WWW-based applications
Impact of Networks

- online communication networks and social media
Impact of Networks

- online communication networks and social media

- implications in
  - knowledge creation
Impact of Networks

- online communication networks and social media

- implications in
  - knowledge creation
  - peoples’ behavior
Impact of Networks

- online communication networks and social media

- implications in
  - knowledge creation
  - peoples’ behavior
  - information sharing
Impact of Networks

- online communication networks and social media
- implications in
  - knowledge creation
  - peoples’ behavior
  - information sharing
  - education
Impact of Networks

- online communication networks and social media

- implications in
  - knowledge creation
  - peoples’ behavior
  - information sharing
  - education
  - democracy
Impact of Networks

- online communication networks and social media

- implications in
  - knowledge creation
  - peoples’ behavior
  - information sharing
  - education
  - democracy
  - society as a whole
Impact of Networks

- 4.76 billion social media users, ca. 60% of the world's population
Impact of Networks

- 4.76 billion social media users, ca. 60% of the world’s population
- 73.3% of all global spending on advertising for digital platforms in 2022
Impact of Networks

- 4.76 billion social media users, ca. 60% of the world’s population
- 73.3% of all global spending on advertising for digital platforms in 2022
- 53.9% of users are concerned about misinformation
Impact of Networks

- 4.76 billion social media users, ca. 60% of the world’s population
- 73.3% of all global spending on advertising for digital platforms in 2022
- 53.9% of users are concerned about misinformation
- Insights can lead to huge monetary and societal impacts
Research Questions in Network Mining

▶ structure discovery
  – communities, summarization, events, role mining
Research Questions in Network Mining

- **structure discovery**
  - communities, summarization, events, role mining

- **study complex dynamic phenomena**
  - evolution, information diffusion, opinion formation, structural prediction, patterns
Research Questions in Network Mining

► structure discovery
  – communities, summarization, events, role mining

► study complex dynamic phenomena
  – evolution, information diffusion, opinion formation, structural prediction, patterns

► develop novel applications and mining primitives
Research Questions in Network Mining

- **structure discovery**
  - communities, summarization, events, role mining

- **study complex dynamic phenomena**
  - evolution, information diffusion, opinion formation, structural prediction, patterns

- develop **novel applications and mining primitives**

- design **efficient algorithms**
Network Mining: Traditional View

- networks represented as pure graph-theory objects
  - no additional vertex / edge information
Network Mining: Traditional View

- networks represented as pure graph-theory objects
  - no additional vertex / edge information
- emphasis on static networks
Temporal Networks: A new lens for network mining

- ability to collect and store large volumes of network data
Temporal Networks: A new lens for network mining

- ability to collect and store large volumes of network data
- available data have time granularity
Temporal Networks: A new lens for network mining

- ability to collect and store large volumes of network data
- available data have time granularity
- lots of additional information associated to vertices/edges
Temporal Networks: A new lens for network mining

- ability to collect and store large volumes of network data
- available data have time granularity
- lots of additional information associated to vertices/edges
- capturing activity and interaction occurring over systems
Temporal Networks: A new lens for network mining

- ability to collect and store large volumes of network data
- available data have time granularity
- lots of additional information associated to vertices/edges
- capturing activity and interaction occurring over systems
- giving rise to new concepts, new problems, and new computational challenges and opportunities
Modeling Activity in Networks

1. network nodes perform actions (e.g., posting messages)
Modeling Activity in Networks

1. network nodes perform actions (e.g., posting messages)

2. network nodes interact with each other
   (e.g., a “like”, a repost, or sending a message to each other)
Many Novel and Interesting Concepts

new pattern types

new types of events

temporal information paths

network evolution
Temporal Network Mining — Objectives

- identify new phenomena to be captured
Temporal Network Mining — Objectives

▶ identify new phenomena to be captured

▶ formulate suitable problems capturing the inherent complexity
Temporal Network Mining — Objectives

- identify new phenomena to be captured
- formulate suitable problems capturing the inherent complexity
- develop algorithmic approaches
Temporal Network Mining — Objectives

- identify new phenomena to be captured
- formulate suitable problems capturing the inherent complexity
- develop algorithmic approaches
- analyze real-world data and gain novel insights
Terminology

we use term “temporal networks”, but terminology is not standardized
Terminology

- we use term “temporal networks”, but terminology is not standardized
- term “X Y” can be encountered in the literature, where

- X:
  - temporal
  - dynamic
  - (time-)evolving
  - time-varying
  - time-dependent
  - evolutionary

- Y:
  - networks
  - graphs

- some combinations have distinct meaning, but not always
Terminology

- we use term “temporal networks”, but terminology is not standardized
- term “$X \ Y$” can be encountered in the literature, where

  $X$: temporal, dynamic, (time-)evolving, time-varying, time-dependent, evolutionary

  $Y$: networks, graphs
Terminology

- we use term “temporal networks”, but terminology is not standardized
- term “X Y” can be encountered in the literature, where

\[ X: \]
- temporal
- dynamic
- (time-)evolving
- time-varying
- time-dependent
- evolutionary

\[ Y: \]
- networks
- graphs

- some combinations have distinct meaning, but not always
Examples of Temporal Networks

- online communication networks
  - phone, email, text messages, etc.

(Holme, 2015)
Examples of Temporal Networks

▷ online communication networks
  - phone, email, text messages, etc.

▷ economic networks
  - credit card transactions
  - trade networks of countries
  - bitcoin transactions

(Holme, 2015)
Examples of Temporal Networks

- **online communication networks**
  - phone, email, text messages, etc.
- **economic networks**
  - credit card transactions
  - trade networks of countries
  - bitcoin transactions
- **bibliographic networks**
  - collaboration and citation networks

(Holme, 2015)
Examples of Temporal Networks

- **human proximity networks**
  - recorded by various sensors and devices, e.g., bluetooth, wifi, etc.
  - patient-referral networks, i.e., how patients are transferred between wards of a hospital system
  - sexual contact networks

(Holme, 2015)
Examples of Temporal Networks

- **human proximity networks**
  - recorded by various sensors and devices, e.g., bluetooth, wifi, etc.
  - patient-referral networks, i.e., how patients are transferred between wards of a hospital system
  - sexual contact networks

- **travel and transportation networks**
  - airline connections, bus transport, bike-sharing systems

(Holme, 2015)
Examples of Temporal Networks

▶ brain networks
  - temporal correlations of the oxygen levels of brain regions as measured by fMRI scanning

(Holme, 2015)
Examples of Temporal Networks

- **brain networks**
  - temporal correlations of the oxygen levels of brain regions as measured by fMRI scanning

- **biological networks**
  - genes involved in different interactions that change over time

(Holme, 2015)
Examples of Temporal Networks

- **brain networks**
  - temporal correlations of the oxygen levels of brain regions as measured by fMRI scanning

- **biological networks**
  - genes involved in different interactions that change over time

- **animal proximity networks**
  - obtained via RFID devices
  - livestock or wildlife

(Holme, 2015)
Representation of Temporal Networks

1. Sequence of interactions
   - A temporal network is represented as $G = (V, E)$

   - This is a lossless representation
   - Also known as sequence of contacts, or sequence of (temporal) edges or temporal edge stream

   - Usually, edges given in chronological order
1. **Sequence of interactions**

- a temporal network is represented as $G = (V, E)$
  - with (static) set of nodes $V$, and
1. Sequence of interactions
   
   a temporal network is represented as $G = (V, E)$
   
   - with (static) set of nodes $V$, and
   
   - set of edges $E = \{(u, v, t)\}$, with $u, v \in V$ and $t \in T$ where $T$ is the time domain
1. **Sequence of interactions**

- a temporal network is represented as $G = (V, E)$
  - with (static) set of nodes $V$, and
  - set of edges $E = \{(u, v, t)\}$, with $u, v \in V$ and $t \in T$ where $T$ is the time domain
  - $T$ is usually $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$
1. **Sequence of interactions**

- a temporal network is represented as $G = (V, E)$
  - with (static) set of nodes $V$, and
  - set of edges $E = \{(u, v, t)\}$, with $u, v \in V$ and $t \in T$ where $T$ is the time domain
  - $T$ is usually $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$
  - if interactions have duration, then $E = \{(u, v, t, \lambda)\}$ with $\lambda$ in $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$
1. Sequence of interactions

- a temporal network is represented as $G = (V, E)$
  - with (static) set of nodes $V$, and
  - set of edges $E = \{(u, v, t)\}$, with $u, v \in V$ and $t \in T$ where $T$ is the time domain
  - $T$ is usually $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$
  - if interactions have duration, then $E = \{(u, v, t, \lambda)\}$ with $\lambda$ in $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$

- this is a lossless representation
Representation of Temporal Networks

1. Sequence of interactions

- a temporal network is represented as $G = (V, E)$
  - with (static) set of nodes $V$, and
  - set of edges $E = \{(u, v, t)\}$, with $u, v \in V$ and $t \in T$ where $T$ is the time domain
  - $T$ is usually $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$
  - if interactions have duration, then $E = \{(u, v, t, \lambda)\}$ with $\lambda$ in $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$

- this is a lossless representation

- also known as sequence of contacts, or sequence of (temporal) edges or temporal edge stream

- usually, edges given in chronological order
Representation of Temporal Networks

1. Sequence of interactions

- visual representation of a temporal network as a sequence of interactions
2. **Sequence of static graphs**

- sequence $G_1, \ldots, G_T$
2. **Sequence of static graphs**

► sequence $G_1, \ldots, G_T$
  - where $G_t = (V_t, E_t)$, with $t = 1, \ldots, T$ with $T \in \mathbb{N}$
2. Sequence of static graphs

- sequence $G_1, \ldots, G_T$
  - where $G_t = (V_t, E_t)$, with $t = 1, \ldots, T$ with $T \in \mathbb{N}$
  - typically assume that nodes are fixed, i.e., $V_t = V$
2. Sequence of static graphs

- sequence $G_1, \ldots, G_T$
  - where $G_t = (V_t, E_t)$, with $t = 1, \ldots, T$ with $T \in \mathbb{N}$
  - typically assume that nodes are fixed, i.e., $V_t = V$
  - $E_t$ are the edges that occur in time interval $t$
2. Sequence of static graphs

- sequence $G_1, \ldots, G_T$
  - where $G_t = (V_t, E_t)$, with $t = 1, \ldots, T$ with $T \in \mathbb{N}$
  - typically assume that nodes are fixed, i.e., $V_t = V$
  - $E_t$ are the edges that occur in time interval $t$

- not equivalent representation with sequence of interactions
Representation of Temporal Networks

2. Sequence of static graphs

- sequence $G_1, \ldots, G_T$
  - where $G_t = (V_t, E_t)$, with $t = 1, \ldots, T$ with $T \in \mathbb{N}$
  - typically assume that nodes are fixed, i.e., $V_t = V$
  - $E_t$ are the edges that occur in time interval $t$

- not equivalent representation with sequence of interactions
  - representation depends on quantization parameter, e.g., seconds, minutes, hours, days, etc.
Representation of Temporal Networks

2. Sequence of static graphs

▶ visual representation of a temporal network as a sequence of static graphs
Time Granularity

- choosing the right time resolution is important

- quantization: binning of time stamps into time intervals of fixed size, e.g., seconds, minutes, hours, days
- coarse resolution may lead to information loss, dense time steps
- fine resolution captures more information but may lead to sparse (or even empty) time steps

- time point in contact network with time resolution of 24h, 1h, and 5 minutes (Lehmann, 2019)

- mean degrees for different time resolutions (Clauset and Eagle, 2012)
Time Granularity

- choosing the right time resolution is important
- quantization: binning of time stamps into time intervals of fixed size, e.g., seconds, minutes, hours, days
Time Granularity

- choosing the right time resolution is important

- quantization: binning of time stamps into time intervals of fixed size, e.g., seconds, minutes, hours, days
  - coarse resolution may lead to information loss, dense time steps
Time Granularity

- choosing the right time resolution is important
- quantization: binning of time stamps into time intervals of fixed size, e.g., seconds, minutes, hours, days
  - coarse resolution may lead to information loss, dense time steps
  - fine resolution captures more information but may lead to sparse (or even empty) time steps
Time Granularity

- choosing the right time resolution is important
- **quantization**: binning of time stamps into time intervals of fixed size, e.g., seconds, minutes, hours, days
  - coarse resolution may lead to information loss, dense time steps
  - fine resolution captures more information but may lead to sparse (or even empty) time steps

![Graphs showing contact network at different time resolutions](image)

- time point in contact network with time resolution of 24h, 1h, and 5 minutes (Lehmann, 2019)
- mean degrees for different time resolutions (Clauset and Eagle, 2012)
Representation of Temporal Networks

3. Time series of contacts
   - a time-series for each pair of nodes in the network
   - equivalent representation with sequence of interactions
Representation of Temporal Networks

3. Time series of contacts
   - a time-series for each pair of nodes in the network
   - equivalent representation with sequence of interactions

4. Tensor representation
   – tensor representing node $\times$ node $\times$ time information
   – can apply powerful tensor-algebra techniques
   – a complication is that time is directed, while tensor algebra assumes that indices can be relabeled (breaking the time ordering)
   – equivalent representation with sequence of interactions
5. **Time-varying graphs** defined as $G = (V, E, T, p, \lambda)$, where
   - $V$: set of nodes
   - $E \subseteq V \times V$: set of edges
   - $T$: a time domain
   - $p: E \times T \rightarrow \{0, 1\}$: a presence function
   - $\lambda: E \times T \rightarrow \mathbb{R}$: a latency function

▶ **equivalent** representation with sequence of interactions
6. **Stream graphs and link streams**

- a formalization for modeling interactions over time
- a stream graph is defined as $G = (T, V, W, E)$, where
  - $T$: a time domain
  - $V$: a set of nodes
  - $W \subseteq T \times V$: a set of temporal nodes
  - $E \subseteq T \times V \times V$: a set of links

  s.t., $(t, u, v) \in E$ implies $(t, u) \in W$ and $(t, v) \in W$

- stream graph: nodes are temporal too

- link stream **equivalent** representation with **sequence of interactions**
Representation of Temporal Networks

(Holme, 2015)

7. Time window graph or underlying graph or projected graph
Representation of Temporal Networks

7. Time window graph or underlying graph or projected graph

- static graph $G_I = (V_I, E_I)$ for given time interval (or window) $I = [a, b]$
7. Time window graph or underlying graph or projected graph

- static graph \( G_I = (V_I, E_I) \) for given time interval (or window) \( I = [a, b] \)
  - with \( E_I = \{ (u, v) \mid (u, v, t) \in E \text{ and } t \in I \} \) be the set of edges with time stamp \( t \in I \)
Representation of Temporal Networks

(Holme, 2015)

7. Time window graph or underlying graph or projected graph

- static graph $G_I = (V_I, E_I)$ for given time interval (or window) $I = [a, b]$ with $E_I = \{(u, v) \mid (u, v, t) \in E \text{ and } t \in I\}$ be the set of edges with time stamp $t \in I$
  - and with $V_I = \{u, v \mid (u, v) \in E_I\}$
Representation of Temporal Networks

7. **Time window graph** or **underlying graph** or **projected graph**

- **static graph** $G_I = (V_I, E_I)$ for given time interval (or window) $I = [a, b]$
  - with $E_I = \{(u, v) \mid (u, v, t) \in E \text{ and } t \in I\}$ be the set of edges with time stamp $t \in I$
  - and with $V_I = \{u, v \mid (u, v) \in E_I\}$

- **time window can be complete time span of** $G$
Representation of Temporal Networks

7. Time window graph or underlying graph or projected graph

- static graph $G_I = (V_I, E_I)$ for given time interval (or window) $I = [a, b]$
  - with $E_I = \{(u, v) \mid (u, v, t) \in E \text{ and } t \in I\}$ be the set of edges with time stamp $t \in I$
  - and with $V_I = \{u, v \mid (u, v) \in E_I\}$

- time window can be complete time span of $G$

- possible to add weights to the static edges, reflecting different temporal aspects, e.g., number of contacts, or sum of durations of contacts
7. Time window graph or underlying graph or projected graph

- static graph $G_I = (V_I, E_I)$ for given time interval (or window) $I = [a, b]$
  - with $E_I = \{(u, v) \mid (u, v, t) \in E$ and $t \in I\}$ be the set of edges with time stamp $t \in I$
  - and with $V_I = \{u, v \mid (u, v) \in E_I\}$

- time window can be complete time span of $G$

- possible to add weights to the static edges, reflecting different temporal aspects, e.g., number of contacts, or sum of durations of contacts

- not equivalent representation with sequence of interactions
Representation of Temporal Networks

7. Time window graph or underlying graph or projected graph

- static graph \( G_I = (V_I, E_I) \) for given time interval (or window) \( I = [a, b] \)
  
  - with \( E_I = \{ (u, v) \mid (u, v, t) \in E \text{ and } t \in I \} \) be the set of edges with time stamp \( t \in I \)
  
  - and with \( V_I = \{ u, v \mid (u, v) \in E_I \} \)

- time window can be complete time span of \( G \)

- possible to add weights to the static edges, reflecting different temporal aspects, e.g., number of contacts, or sum of durations of contacts

- not equivalent representation with sequence of interactions

- usually results in loss of information
Representation of Temporal Networks

7. time window graph or underlying graph or projected graph

time window graphs for intervals [1, 9], [4, 9], [6, 7]
Temporal Graph Variants

- time-intervals instead of time stamps
- directed vs. undirected edges
- multi edges
- (time-variant) labeled or colored nodes and edges
- (time-variant) node and edge features
- temporal hypergraphs

Combinations possible: temporal multi-layer hypergraphs with node features
Temporal Networks vs. Dynamic Graphs

- **dynamic graphs** is a standard model typically studied in *theoretical computer science* – e.g., (Henzinger et al., 1999; Thorup, 2000; Hanauer et al., 2021)

- dynamic graphs are represented as a sequence of *edge additions and/or edge deletions*
Temporal Networks vs. Dynamic Graphs

- **dynamic graphs** is a standard model typically studied in **theoretical computer science**
  - e.g., (Henzinger et al., 1999; Thorup, 2000; Hanauer et al., 2021)

- **dynamic graphs** are represented as a **sequence of edge additions and/or edge deletions**

- **objective**: efficient maintenance of graph properties
  - e.g., connectivity, shortest paths, spanners, etc.
Temporal Networks vs. Dynamic Graphs

- **dynamic graphs** is a standard model typically studied in **theoretical computer science**
  - e.g., (Henzinger et al., 1999; Thorup, 2000; Hanauer et al., 2021)

- dynamic graphs are represented as a **sequence of edge additions and/or edge deletions**

- **objective**: efficient maintenance of graph properties
  - e.g., connectivity, shortest paths, spanners, etc.

- emphasis on **computational efficiency**
  - computation time **per operation**
  - e.g., cost of maintaining a minimum spanning tree per edge additions/deletions
Temporal Networks vs. Dynamic Graphs

- dynamic graphs resemble sequence of interactions model
Temporal Networks vs. Dynamic Graphs

- **dynamic graphs resemble sequence of interactions** model
- main difference lies on which **graph properties** we study
- for dynamic graphs we typically consider **properties on graph snapshots**
  - i.e., minimum spanning tree on the current snapshot
Temporal Networks vs. Dynamic Graphs

- **dynamic graphs resemble sequence of interactions** model
- main difference lies on which **graph properties** we study
- for dynamic graphs we typically consider **properties on graph snapshots**
  - i.e., minimum spanning tree on the current snapshot
- for temporal graphs we typically consider **properties that span a time interval**
  - i.e., a temporal pattern
Temporal Networks vs. Dynamic Graphs

- **dynamic graphs resemble sequence of interactions model**
- main difference lies on which **graph properties** we study
- for dynamic graphs we typically consider **properties on graph snapshots**
  - i.e., minimum spanning tree on the current snapshot
- for temporal graphs we typically consider **properties that span a time interval**
  - i.e., a temporal pattern

- **_disclaimer:_ we do not consider_** dynamic graphs
Dynamic Networks in Network Generation

- in graph generation models, we consider dynamic networks
  - e.g., Barabási-Albert, forest-fire, copying model, etc.
Dynamic Networks in Network Generation

- in graph generation models, we consider dynamic networks
  - e.g., Barabási-Albert, forest-fire, copying model, etc.
- similar to dynamic graphs: a sequence of node/edge additions
  (typically no deletions)

![Network Diagrams]
Dynamic Networks in Network Generation

- In graph generation models, we consider dynamic networks
  - E.g., Barabási-Albert, forest-fire, copying model, etc.

- Similar to dynamic graphs: a sequence of node/edge additions (typically no deletions)

- Node/edge addition are governed by a probabilistic model, not arbitrary, or worst case, as in algorithmic models
in graph generation models, we consider dynamic networks
  – e.g., Barabási-Albert, forest-fire, copying model, etc.

similar to dynamic graphs: a sequence of node/edge additions
  (typically no deletions)

node/edge addition are governed by a probabilistic model, not
  arbitrary, or worst case, as in algorithmic models

emphasis again on network topology, i.e., how certain network
  structures emerge
▶ in graph generation models, we consider dynamic networks
  – e.g., Barabási-Albert, forest-fire, copying model, etc.
▶ similar to dynamic graphs: a sequence of node/edge additions (typically no deletions)
▶ node/edge addition are governed by a probabilistic model, not arbitrary, or worst case, as in algorithmic models
▶ emphasis again on network topology, i.e., how certain network structures emerge
  – e.g., scale-free distribution, small world, etc.

Dynamic Networks in Network Generation
Dynamic Networks in Network Generation

- in graph generation models, we consider dynamic networks
  - e.g., Barabási-Albert, forest-fire, copying model, etc.
- similar to dynamic graphs: a sequence of node/edge additions (typically no deletions)
- node/edge addition are governed by a probabilistic model, not arbitrary, or worst case, as in algorithmic models
- emphasis again on network topology, i.e., how certain network structures emerge
  - e.g., scale-free distribution, small world, etc.
- disclaimer: we do not consider dynamic networks in this context
Graph Streams

▶ data-stream model: (Muthukrishnan et al., 2005)

– data are presented as a sequence of data items (potentially infinite)
– assume a small number of passes, typically constant or just one pass
– assume small memory compared to data size, e.g., poly-logarithmic
– assume fast computation per data item processed, e.g., constant or poly-logarithmic

▶ a graph stream is a graph dataset in the data-stream model
  e.g., sequence of interactions (temporal network), or edge additions/deletions (dynamic graph)

▶ a graph stream is not a representation model, but underlying computational model

▶ we can study questions of mining temporal networks in the graph-stream model
Graph Streams

- data-stream model: (Muthukrishnan et al., 2005)
  - data are presented as a sequence of data items (potentially infinite)
Graph Streams

▶ data-stream model: (Muthukrishnan et al., 2005)
  - data are presented as a sequence of data items (potentially infinite)
  - assume a small number of passes, typically constant or just one pass
Graph Streams

- **data-stream model:**
  - data are presented as a sequence of data items (potentially infinite)
  - assume a small number of passes, typically constant or just one pass
  - assume small memory compared to data size, e.g., poly-logarithmic

(Muthukrishnan et al., 2005)
Graph Streams

- data-stream model: (Muthukrishnan et al., 2005)
  - data are presented as a sequence of data items (potentially infinite)
  - assume a small number of passes, typically constant or just one pass
  - assume small memory compared to data size, e.g., poly-logarithmic
  - assume fast computation per data item processed, e.g., constant or poly-logarithmic
Graph Streams

- data-stream model:
  - data are presented as a sequence of data items (potentially infinite)
  - assume a small number of passes, typically constant or just one pass
  - assume small memory compared to data size, e.g., poly-logarithmic
  - assume fast computation per data item processed, e.g., constant or poly-logarithmic

- a graph stream is a graph dataset in the data-stream model

(Muthukrishnan et al., 2005)
Graph Streams

- data-stream model:
  - data are presented as a sequence of data items (potentially infinite)
  - assume a small number of passes, typically constant or just one pass
  - assume small memory compared to data size, e.g., poly-logarithmic
  - assume fast computation per data item processed, e.g., constant or poly-logarithmic

- a graph stream is a graph dataset in the data-stream model
  e.g., sequence of interactions (temporal network), or edge additions/deletions (dynamic graph)

(Muthukrishnan et al., 2005)
Graph Streams

- **data-stream model:**
  - data are presented as a sequence of data items (potentially infinite)
  - assume a small number of passes, typically constant or just one pass
  - assume small memory compared to data size, e.g., poly-logarithmic
  - assume fast computation per data item processed, e.g., constant or poly-logarithmic

- a graph stream is a graph dataset in the data-stream model
  e.g., sequence of interactions (temporal network), or edge additions/deletions (dynamic graph)

- a graph stream is not a representation model, but underlying computational model
Graph Streams

- **data-stream model:**
  - data are presented as a **sequence of data items** (potentially infinite)
  - assume a **small number of passes**, typically constant or just one pass
  - assume **small memory** compared to data size, e.g., poly-logarithmic
  - assume **fast computation** per data item processed, e.g., constant or poly-logarithmic

- a **graph stream** is a **graph dataset** in the **data-stream model**
  - e.g., sequence of interactions (temporal network), or edge additions/deletions (dynamic graph)

- a graph stream is **not a representation model**, but underlying **computational model**

- we can study questions of mining temporal networks in the graph-stream model
Temporal Graph Learning

- rich and fast growing body of works on temporal graph learning
Temporal Graph Learning

- rich and fast growing body of works on temporal graph learning
- tasks: dynamic link/node property prediction, graph classification, clustering, link prediction, representation learning, ...
Temporal Graph Learning

- **rich and fast growing** body of works on temporal graph learning

- **tasks**: dynamic link/node property prediction, graph classification, clustering, link prediction, representation learning, ...

- **methods**: graph neural networks, recurrent neural network, graph transformers, ...
Temporal Graph Learning

- rich and fast growing body of works on temporal graph learning
- **tasks**: dynamic link/node property prediction, graph classification, clustering, link prediction, representation learning, ...
- **methods**: graph neural networks, recurrent neural network, graph transformers, ...
- several extensive surveys available, e.g.,
  (Gupta and Bedathur, 2022; Longa et al., 2023; Kazemi et al., 2020)
Temporal Graph Learning

- rich and fast growing body of works on temporal graph learning
- tasks: dynamic link/node property prediction, graph classification, clustering, link prediction, representation learning, ...
- methods: graph neural networks, recurrent neural network, graph transformers, ...
- several extensive surveys available, e.g.,
  (Gupta and Bedathur, 2022; Longa et al., 2023; Kazemi et al., 2020)
- disclaimer: in this tutorial we do not consider temporal graph learning
Theoretical Aspects of Temporal Graphs

- how is the complexity of classic combinatorial optimization problems changes when time is added?

- some old results: the max-flow min-cut theorem holds with unit capacities for temporal paths (Berman, 1996)
Theoretical Aspects of Temporal Graphs

- how is the complexity of classic combinatorial optimization problems changes when time is added?

- some old results: the max-flow min-cut theorem holds with unit capacities for temporal paths (Berman, 1996)

- a number of recent works
  - graph coloring
  - maximal matching
  - cliques
  - network design
  - path problems
  - vertex cover
  - ...

- discussing complexity, FPT algorithms, enumeration, etc.
Agenda

Part I: Introduction and Motivation
▶ models of temporal networks
▶ algorithmic approaches

Part II: Mining Temporal Networks A:
▶ connectivity, temporal properties
▶ centrality, cores

Part III: Mining Temporal Networks B:
▶ communities, patterns and events
▶ diffusion and random networks

Part IV: Tools and Code Libraries

Part V: Challenges, Open Problems, and Trends
Part II

Mining Temporal Networks A
Time-respecting Walks and Paths

- a fundamental concept in analysis of temporal networks
Time-respecting Walks and Paths

- A fundamental concept in analysis of temporal networks.
- A time-respecting (or temporal) walk is a sequence of temporal edges, such that:
  - Consecutive edges share a common node.
  - Time stamps of temporal edges are increasing.
  - (Non-strict version: time stamps non-decreasing.)
Time-respecting Walks and Paths

- a fundamental concept in analysis of temporal networks
- a time-respecting (or temporal) walk is a sequence of temporal edges, such that
  - consecutive edges share a common node, and
Time-respecting Walks and Paths

- a **fundamental concept** in analysis of temporal networks

- a **time-respecting (or temporal) walk** is a sequence of temporal edges, such that
  - consecutive edges **share a common node**, and
  - time stamps of temporal edges are **increasing**
Time-respecting Walks and Paths

- a fundamental concept in analysis of temporal networks
- a time-respecting (or temporal) walk is a sequence of temporal edges, such that
  - consecutive edges share a common node, and
  - time stamps of temporal edges are increasing
  - (non-strict version: time stamps non-decreasing)
Time-respecting Walks and Paths

- a fundamental concept in analysis of temporal networks
- a time-respecting (or temporal) walk is a sequence of temporal edges, such that
  - consecutive edges share a common node, and
  - time stamps of temporal edges are increasing
  - (non-strict version: time stamps non-decreasing)
- a temporal paths is a temporal walk that visits each vertex at most once
Time-respecting Walks and Paths

- a fundamental concept in analysis of temporal networks
- a time-respecting (or temporal) walk is a sequence of temporal edges, such that
  - consecutive edges share a common node, and
  - time stamps of temporal edges are increasing
  - (non-strict version: time stamps non-decreasing)
- a temporal paths is a temporal walk that visits each vertex at most once
Time-respecting Paths — Example

\[(c, e, 2), (e, d, 5), (d, b, 6)\] is a time-respecting path from \(c\) to \(b\)
(c, e, 2), (e, d, 5), (d, b, 6) is a time-respecting path from c to b

(c, b, 3), (b, a, 1) is not a time-respecting path
**Time-respecting Paths — Example**

(c, e, 2), (e, d, 5), (d, b, 6) is a time-respecting path from c to b

(c, b, 3), (b, a, 1) is not a time-respecting path

▶ **non-symmetric**: from e to b but not from b to e
Time-respecting Paths — Example

(c, e, 2), (e, d, 5), (d, b, 6) is a time-respecting path from c to b

(c, b, 3), (b, a, 1) is not a time-respecting path

▶ non-symmetric: from e to b but not from b to e
▶ non-transitive: from b to d and from d to e but not from b to e
Applications

- information (or disease) can only propagate over time-respecting walks
Applications

- Information (or disease) can only propagate over \textit{time-respecting walks}.

- Communication networks: capture possible flow of information.
Applications

▶ information (or disease) can only propagate over time-respecting walks

▶ communication networks: capture possible flow of information

▶ financial networks: trace the sequence of financial exchanges to identify patterns, detect fraudulent activity, or assess market dynamics
Applications

- Information (or disease) can only propagate over time-respecting walks.
- Communication networks: capture possible flow of information.
- Financial networks: trace the sequence of financial exchanges to identify patterns, detect fraudulent activity, or assess market dynamics.
- Epidemiology: understanding the spread of diseases.
Applications

- information (or disease) can only propagate over time-respecting walks

- communication networks: capture possible flow of information

- financial networks: trace the sequence of financial exchanges to identify patterns, detect fraudulent activity, or assess market dynamics

- epidemiology: understanding the spread of diseases

- social network analysis: centrality measures for ranking users
Temporal Reachability

Reachability is defined as in static graphs, but using time-respecting walks
Temporal Reachability

Reachability is defined as in static graphs, but using time-respecting walks

"There exists temporal \((s, z)\)-walk in \(G\)" not always useful—the existence depends on time interval
Temporal Reachability

Reachability is defined as in static graphs, but using time-respecting walks.

"There exists temporal \((s, z)\)-walk in \(G\)" not always useful—the existence depends on time interval.
Temporal Reachability

Reachability is defined as in static graphs, but using time-respecting walks

"There exists temporal \( (s, z) \)-walk in \( G \)" not always useful—the existence depends on time interval

- temporal walks and paths problems, and reachability are always related to a time interval
Temporal Reachability

Reachability is defined as in static graphs, but using time-respecting walks

"There exists temporal \((s, z)\)-walk in \(G\)" not always useful—the existence depends on time interval

- temporal walks and paths problems, and reachability are always related to a time interval \(I\)
- if no time-interval given, we take the complete span of \(G\)
Temporal Reachability

Reachability is defined as in static graphs, but using time-respecting walks

"There exists temporal \((s, z)\)-walk in \(G\)“ not always useful—the existence depends on time interval

- temporal walks and paths problems, and reachability are always related to a time interval \(I\)
- if no time-interval given, we take the complete span of \(G\)
- finding all from \(s\) reachable nodes in linear time (later)
A Reachability Problem

Temporal Exploration Problem

- **Given:** Temporal graph \( G = (V, E) \), vertex \( s \in V \)
- **Question:** Can we reach all other nodes with a single temporal walk starting from \( s \)?

Problem is NP-complete (corresponding problem in static graphs in linear time!)

If transition time non-zero \( \Rightarrow \) each edge at most once in a strict temporal walk

Proof idea:
- reduction from Hamilton path problem (Michail and Spirakis, 2016)
A Reachability Problem

Temporal Exploration Problem

- **Given**: Temporal graph \( G = (V, E) \), vertex \( s \in V \)
- **Question**: Can we reach all other nodes with a single temporal walk starting from \( s \)?

Problem is NP-complete  (corresponding problem in static graphs in linear time!)

If transition time non-zero \( \Rightarrow \) each edge at most once in a strict temporal walk
A Reachability Problem

Temporal Exploration Problem

- **Given**: Temporal graph $G = (V, E)$, vertex $s \in V$
- **Question**: Can we reach all other nodes with a single temporal walk starting from $s$?

**Problem is NP-complete**  (corresponding problem in static graphs in linear time!)

If transition time non-zero $\Rightarrow$ each edge at most once in a strict temporal walk

Proof idea:
- reduction from Hamilton path problem

(Michail and Spirakis, 2016)
Connectivity and Connected Components

- **static connected components** are based on reachability
  - each node in connected component \( C \) can reach each all other nodes in \( C \)
- **equivalence relation that partitions the graph**
  - reflexivity, symmetry, transitivity
Connectivity and Connected Components

- **static connected components** are based on reachability
  - each node in connected component $C$ can reach each all other nodes in $C$
- equivalence relation that partitions the graph
  - reflexivity, symmetry, transitivity
- **temporal connected components** are based on temporal reachability
  - a subset of the nodes $C \subseteq V$
  - there is a temporal walk between each pair $u, v \in C$
Connectivity and Connected Components

**Temporal Connectivity Problem**

- **Given:** A temporal graph and integer $k$
- **Question:** Is there a subset of the vertices $V' \subseteq V$ of size $k$ such that all vertices in $V'$ can reach each other by a temporal path?

- two versions: **open variant** allows paths using nodes outside of $V'$, **closed variant** not
- both cases are **NP-complete** (Bhadra and Ferreira, 2003)
Connectivity and Connected Components

**Temporal Connectivity Problem**

- **Given:** A temporal graph and integer $k$
- **Question:** Is there a subset of the vertices $V' \subseteq V$ of size $k$ such that all vertices in $V'$ can reach each other by a temporal path?

- Two versions: **open variant** allows paths using nodes outside of $V'$, **closed variant** not
- Both cases are **NP-complete** (Bhadra and Ferreira, 2003)

Connected components can be overlapping

- $a$ and $d$ are openly connected
Time Window Reachability and Connectivity

- compute reachability or connected components in time window graph $G_I = (V_I, E_I)$
  - given time window $I = [a, b]$, $G_I = (V_I, E_I)$ is static graph induced by edges appearing in $I$
Time Window Reachability and Connectivity

- compute reachability or connected components in time window graph $G_I = (V_I, E_I)$
  - given time window $I = [a, b]$, $G_I = (V_I, E_I)$ is static graph induced by edges appearing in $I$

- time window reachability for $u \in V$ in time interval $I = [a, b]$ (Wen et al., 2020)
  - the set of nodes that $u$ can reach in $G_I = (V_I, E_I)$ with static walk

- both problems solvable in linear time

- alternative: index-based algorithms for large scale graphs
Time Window Reachability and Connectivity

- compute reachability or connected components in time window graph $G_I = (V_I, E_I)$
  - given time window $I = [a, b]$, $G_I = (V_I, E_I)$ is static graph induced by edges appearing in $I$

- time window reachability for $u \in V$ in time interval $I = [a, b]$ (Wen et al., 2020)
  - the set of nodes that $u$ can reach in $G_I = (V_I, E_I)$ with static walk

- time window connected components in time interval $I = [a, b]$ (Xie et al., 2023)
  - the set of static connected components in $G_I = (V_I, E_I)$
Time Window Reachability and Connectivity

- compute reachability or connected components in time window graph \( G_I = (V_I, E_I) \)
  - given time window \( I = [a, b] \), \( G_I = (V_I, E_I) \) is static graph induced by edges appearing in \( I \)

- time window reachability for \( u \in V \) in time interval \( I = [a, b] \) (Wen et al., 2020)
  - the set of nodes that \( u \) can reach in \( G_I = (V_I, E_I) \) with static walk

- time window connected components in time interval \( I = [a, b] \) (Xie et al., 2023)
  - the set of static connected components in \( G_I = (V_I, E_I) \)

- both problems solvable in linear time
- alternative: index-based algorithms for large scale graphs
some paths in the static graph are not meaningful in the temporal graph

(Wu et al., 2014)
some paths in the static graph are not meaningful in the temporal graph

- e.g., $a - b - g - j$ is not time-respecting path
some paths in the static graph are not meaningful in the temporal graph
- e.g., $a \rightarrow b \rightarrow g \rightarrow j$ is not time-respecting path
- what is an optimal path from $a$ to $k$?

(Wu et al., 2014)
Minimum Temporal Paths

- **earliest-arrival path**: a path from \( x \) to \( y \) with earliest arrival time

(Wu et al., 2014)
Minimum Temporal Paths

- **earliest-arrival path**: a path from $x$ to $y$ with earliest arrival time
- **latest-departure path**: a path from $x$ to $y$ with latest departure time

(Wu et al., 2014)
Minimum Temporal Paths

- **earliest-arrival path**: a path from \( x \) to \( y \) with earliest arrival time
- **latest-departure path**: a path from \( x \) to \( y \) with latest departure time
- **fastest path**: path from \( x \) to \( y \) with minimum elapsed time

(Wu et al., 2014)
Minimum Temporal Paths

- **earliest-arrival path**: a path from \( x \) to \( y \) with earliest arrival time
- **latest-departure path**: a path from \( x \) to \( y \) with latest departure time
- **fastest path**: path from \( x \) to \( y \) with minimum elapsed time
- **shortest path**: a path from \( x \) to \( y \) minimum sum of traversal times

(Wu et al., 2014)
Minimum Temporal Paths

- **earliest-arrival path**: a path from \( x \) to \( y \) with earliest arrival time
- **latest-departure path**: a path from \( x \) to \( y \) with latest departure time
- **fastest path**: path from \( x \) to \( y \) with minimum elapsed time
- **shortest path**: a path from \( x \) to \( y \) minimum sum of traversal times
- **minimum hop path**: a path from \( x \) to \( y \) with minimum number of hops

\[(Wu \ et \ al., \ 2014)\]
Let $P_{(s,z)}$ be an optimal temporal path and $P$ a subpath of $P_{(s,z)}$, then, in general, $P$ is not optimal.

Example:
- fastest path $(s, d, b, z)$ with duration 4
**Optimal Path Computation**

**Observation**

Let $P_{(s,z)}$ be an optimal temporal path and $P$ a subpath of $P_{(s,z)}$, then, in general, $P$ is not optimal.

**Example:**
- fastest path $(s, d, b, z)$ with duration 4
- subpath $(s, d, b)$ has duration 3 but path $(s, a, b)$ has duration 2
Optimal Path Computation

**Observation**

Let $P_{(s,z)}$ be an optimal temporal path and $P$ a subpath of $P_{(s,z)}$, then, in general, $P$ is not optimal.

**Example:**

- fastest path $(s, d, b, z)$ with duration 4
- subpath $(s, d, b)$ has duration 3 but path $(s, a, b)$ has duration 2
- similar examples for other variants
**Optimal Path Computation**

**Observation**
Let $P_{(s,z)}$ be an optimal temporal path and $P$ a subpath of $P_{(s,z)}$, then, in general, $P$ is not optimal.

**Example:**
- fastest path $(s, d, b, z)$ with duration 4
- subpath $(s, d, b)$ has duration 3 but path $(s, a, b)$ has duration 2
- similar examples for other variants
- greedy Dijkstra does not work in general
Earliest-arrival Path

(Wu et al., 2014)

**Subpath Optimality**

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path.
Earliest-arrival Path

(Wu et al., 2014)

**Subpath Optimality**

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path

- source vertex \(x\), starting time \(t_s\)
Earliest-arrival Path

(Wu et al., 2014)

Subpath Optimality

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path

- source vertex \(x\), starting time \(t_s\)
- array \(A\) of size \(|V|\) to record arrival times to each node
Earliest-arrival Path

(Wu et al., 2014)

Subpath Optimality

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path

- source vertex \(x\), starting time \(t_s\)
- array \(A\) of size \(|V|\) to record arrival times to each node
- \(A[x] = t_s\) and \(A[v] = \infty\), for nodes other than source
Earliest-arrival Path

(Wu et al., 2014)

**Subpath Optimality**

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path.

- source vertex \(x\), starting time \(t_s\)
- array \(A\) of size \(|V|\) to record arrival times to each node
- \(A[x] = t_s\) and \(A[v] = \infty\), for nodes other than source
- process edges \((u, v, t, \lambda)\) in temporal order
Earliest-arrival Path

(Wu et al., 2014)

Subpath Optimality

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path

- source vertex \(x\), starting time \(t_s\)
- array \(A\) of size \(|V|\) to record arrival times to each node
- \(A[x] = t_s\) and \(A[v] = \infty\), for nodes other than source
- process edges \((u, v, t, \lambda)\) in temporal order
  - if \(t \geq A[u]\) (\(u\) is already reached from \(x\))
Earliest-arrival Path

(Wu et al., 2014)

**Subpath Optimality**

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path.

- **source vertex** \(x\), starting time \(t_s\)
- **array** \(A\) of size \(|V|\) to record arrival times to each node
- \(A[x] = t_s\) and \(A[v] = \infty\), for nodes other than source
- **process edges** \((u, v, t, \lambda)\) in temporal order
  - if \(t \geq A[u]\) \((u\) is already reached from \(x)\)
  - current edge might be earliest path from \(x\) to \(v\)
Earliest-arrival Path

(Wu et al., 2014)

Subpath Optimality

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path

- source vertex \(x\), starting time \(t_s\)
- array \(A\) of size \(|V|\) to record arrival times to each node
- \(A[x] = t_s\) and \(A[v] = \infty\), for nodes other than source
- process edges \((u, v, t, \lambda)\) in temporal order
  - if \(t \geq A[u]\) (\(u\) is already reached from \(x\))
  - current edge might be earliest path from \(x\) to \(v\)
  - update \(A[v] = \min(A[v], t + \lambda)\)
Earliest-arrival Path

(Wu et al., 2014)

Subpath Optimality

If there exists an earliest arrival \((s, z)\)-path, then there exist an earliest arrival \((s, z)\)-path \(P\) such that each prefix path of \(P\) is an earliest arrival path

- source vertex \(x\), starting time \(t_s\)
- array \(A\) of size \(|V|\) to record arrival times to each node
- \(A[x] = t_s\) and \(A[v] = \infty\), for nodes other than source
- process edges \((u, v, t, \lambda)\) in temporal order
  - if \(t \geq A[u]\) \((u\) is already reached from \(x\))
  - current edge might be earliest path from \(x\) to \(v\)
  - update \(A[v] = \min(A[v], t + \lambda)\)

Linear time algorithm
Latest-departure Path

(Wu et al., 2014)

- temporal graph $G = (V, E)$
- sink vertex $x$, ending time $t_s$
Latest-departure Path

- temporal graph $G = (V, E)$
- sink vertex $x$, ending time $t_s$
- same process as for earliest-arrival path, but
Latest-departure Path

- temporal graph $G = (V, E)$
- sink vertex $x$, ending time $t_s$
- same process as for earliest-arrival path, but
- process edges in reversed temporal order

(Wu et al., 2014)
Latest-departure Path

- temporal graph $G = (V, E)$
- sink vertex $x$, ending time $t_s$

- same process as for earliest-arrival path, but
- process edges in reversed temporal order
- add new interaction to the path if it does not violate temporal order
Dominating Path

(Wu et al., 2014)

- source vertex $x$ and sink $v$
- for a path $P_1$ arriving at $v$ let $(a, s)$, where
  - $a$: time of arrival at $v$
  - $s$: time of departure from $x$
- consider another path $P_2$ arriving at $v$ with $(a', s')$
Dominating Path

- source vertex $x$ and sink $v$

- for a path $P_1$ arriving at $v$ let $(a, s)$, where
  - $a$: time of arrival at $v$
  - $s$: time of departure from $x$

- consider another path $P_2$ arriving at $v$ with $(a', s')$

- if $(s' > s$ and $a' \leq a)$ or $(s' = s$ and $a' < a)$
  - then path $P_2$ dominates path $P_1$
  - because $a' - s' < a - s$

(Wu et al., 2014)
Dominating Path

(Wu et al., 2014)

- source vertex \( x \) and sink \( v \)
- for a path \( P_1 \) arriving at \( v \) let \((a, s)\), where
  - \( a \): time of arrival at \( v \)
  - \( s \): time of departure from \( x \)
- consider another path \( P_2 \) arriving at \( v \) with \((a', s')\)

- if \((s' > s \text{ and } a' \leq a)\) or \((s' = s \text{ and } a' < a)\)
  - then path \( P_2 \) dominates path \( P_1 \)
  - because \( a' - s' < a - s \)

- we can replace \( P_1 \) with \( P_2 \) and improve duration
Finding Optimal Path

Fastest Path

- streaming algorithm can be adapted using dominating paths (Wu et al., 2014)
Finding Optimal Path

Fastest Path

Stream the algorithm can be adapted using dominating paths

- keep a list of non-dominated labels at each node

(Wu et al., 2014)
Finding Optimal Path

Fastest Path

- streaming algorithm can be adapted using dominating paths
  - keep a list of non-dominated labels at each node
  - when new edge arrives extend non-dominated path

(Wu et al., 2014)
Finding Optimal Path

Fastest Path

- streaming algorithm can be adapted using dominating paths
  - keep a list of non-dominated labels at each node
  - when new edge arrives extend non-dominated path

Shortest and Minimum Hop Path

- similar to algorithm for fastest path
Finding Optimal Path

Fastest Path

- streaming algorithm can be adapted using *dominating paths*
  - keep a *list of non-dominated labels* at each node
  - when new edge arrives extend *non-dominated path*

Shortest and Minimum Hop Path

- *similar* to algorithm for *fastest path*
  - but keep track of non-dominated path wrt to the
  - *transition times*, or
Finding Optimal Path

Fastest Path

▶ streaming algorithm can be adapted using dominating paths
  – keep a list of non-dominated labels at each node
  – when new edge arrives extend non-dominated path

Shortest and Minimum Hop Path

▶ similar to algorithm for fastest path
  – but keep track of non-dominated path wrt to the
  – transition times, or
  – number of interactions, instead of the duration
Finding Optimal Path

**Fastest Path**

- streaming algorithm can be adapted using *dominating paths*
  - keep a *list of non-dominated labels* at each node
  - when new edge arrives extend non-dominated path

(Wu et al., 2014)

**Shortest and Minimum Hop Path**

- similar to algorithm for *fastest path*
  - but keep track of non-dominated path wrt to the
  - *transition times*, or
  - *number of interactions*, instead of the duration

Linear time algorithms in case of equal transition times
Restless Walks and Paths

- function $\beta : V \rightarrow \mathbb{R}$ determines the maximum waiting time at each node
Restless Walks and Paths

- function $\beta : V \rightarrow \mathbb{R}$ determines the maximum waiting time at each node
  - motivation, e.g., spreading of disease
Restless Walks and Paths

- function $\beta : V \to \mathbb{R}$ determines the maximum waiting time at each node
  - motivation, e.g., spreading of disease

- finding restless temporal paths is NP-hard

(Casteigts et al., 2021)
Restless Walks and Paths

- Function $\beta : V \rightarrow \mathbb{R}$ determines the maximum waiting time at each node
  - Motivation, e.g., spreading of disease

- Finding restless temporal paths is NP-hard (Casteigts et al., 2021)

- Finding restless temporal walks possible in $\mathcal{O}(|V| + |E| \log |E|)$ (Himmel et al., 2019)
Restless Walks and Paths

- function $\beta : V \rightarrow \mathbb{R}$ determines the maximum waiting time at each node
  - motivation, e.g., spreading of disease

- finding restless temporal paths is NP-hard (Casteigts et al., 2021)

- finding restless temporal walks possible in $O(|V| + |E| \log |E|)$ (Himmel et al., 2019)

- extended for colored restless path and reachability (Thejaswi et al., 2020)
Static Expansion of a Temporal Network

- transformation of a temporal network to a directed static network
  - temporal paths in temporal network correspond to static paths in the directed static network

- how to create such a transformation?
Static Expansion of a Temporal Network

- **transformation** of a **temporal network** to a **directed static network**
  - temporal paths in temporal network correspond to static paths in the directed static network

- how to create such a transformation?

1. create a copy of each node for each time instance
Static Expansion of a Temporal Network

- **transformation** of a **temporal network** to a **directed static network**
  - **temporal paths** in temporal network correspond to **static paths** in the directed static network

- how to create such a transformation?

1. create a copy of each node for each time instance
2. create a directed edge from the \((t-1)\)-th copy of \(u\) to the \(t\)-th copy of \(u\), for all nodes \(u\), and all time instances \(t\)
Static Expansion of a Temporal Network

- **transformation** of a **temporal network** to a **directed static network**
  - **temporal paths** in temporal network **correspond** to **static paths** in the **directed static network**

- how to create such a transformation?

1. create a copy of each node for each time instance
2. create a directed edge from the \((t-1)\)-th copy of \(u\) to the \(t\)-th copy of \(u\), for all nodes \(u\), and all time instances \(t\)
3. create directed edges for the temporal edges
Static Expansion Graphs

- $V_e = \{(v, t) \mid v \in V, \ t \in T\}$, where $T$ is the set of all possible timestamps
- edges $E_e$: interactions between the temporal nodes $V_t$
Static Expansion Graphs

- $V_e = \{(v, t) | v \in V, \ t \in T\}$, where $T$ is the set of all possible timestamps
- edges $E_e$ : interactions between the temporal nodes $V_t$

Problem: Size in number of time-steps $\Theta(\mid T\mid \cdot \mid V\mid + \mid E\mid)$
Static Expansion Graphs

- $V_e = \{(v, t) \mid v \in V, \ t \in T(v)\}$, where $T(v)$ is the set of times with activity at $v$
- edges $E_e$ : interactions between the temporal nodes $V_t$
Static Expansion Graphs

- $V_e = \{(v, t) \mid v \in V, \ t \in T(v)\}$, where $T(v)$ is the set of times with activity at $v$
- edges $E_e$ : interactions between the temporal nodes $V_t$

Size in number of edges $O(|E|)$
Directed Line Graph

Temporal graph

Directed line graph

- temporal walk in $G$ of length $\ell + 1 \iff$ walk of length $\ell$ in $D(G)$
- counting walks by matrix powers of adjacency matrix
- size in $O(|E|^2)$
Static Representations

- static expansion graph and directed line graph are directed acyclic graphs if edges have non-zero transition times

- standard graph algorithms (bfs, dfs, Dijkstra, Bellman-Ford) can be adopted for finding
  - optimal temporal paths and temporal walks

- upstream, downstream reachability sets
Transportation Temporal Networks

(Kujala et al., 2018)
Pareto-optimal Journeys

(Kujala et al., 2018)
Pareto-optimal Journeys

Weighted Temporal Graph
- Additional edge costs \((u, v, t, \lambda, c)\) with \(c \in \mathbb{R}\)

Bicriteria optimal paths
- solution: pair (duration, costs)
- non-stop: fast but expensive (2h, 200)
- via Munich: slow but cheap (4h, 100)
Pareto-optimal Journeys

**Weighted Temporal Graph**

- Additional edge costs \((u, v, t, \lambda, c)\) with \(c \in \mathbb{R}\)

**Bicriteria optimal paths**

- solution: pair (duration, costs)

- non-stop: fast but expensive (2h, 200)
- via Munich: slow but cheap (4h, 100)

**Enumeration** of temporal paths that are efficient wrt. duration and cost in polynomial delay and space

(Mutzel and Oettershagen, 2019)
Temporal Graph Properties

- many static graph properties need to be adapted for temporal graphs to be meaningful
- local and global properties, often several variants with different focus

Diameter
- shortest latency of time-respecting paths over connected pairs (Chaintreau et al., 2007)
- restricted on connected pairs, as real data have many disconnected pairs
- the minimum integer \( d \) for which the duration between each pair of nodes \( u, v \in V \) is at most \( d \) (over all possible starting times) (Michail, 2016)
Temporal Graph Properties

- many static graph properties need to be adapted for temporal graphs to be meaningful
- local and global properties, often several variants with different focus

- **diameter**
  - shortest latency of time-respecting paths over connected pairs \( (\text{Chaintreau et al., 2007}) \)
  - restricted on connected pairs, as real data have many disconnected pairs

- the minimum integer \( d \) for which the duration between each pair of nodes \( u, v \in V \) is at most \( d \) (over all possible starting times) \( (\text{Michail, 2016}) \)
Temporal Network Efficiency

- **network efficiency**: the harmonic mean of durations (latency) over all pairs \((\text{Tang et al., 2009})\)

\[
E(t_1, t_2) = \frac{1}{n(n-1)} \sum_{u,v \in V} \frac{1}{d_{(t_1, t_2)}(u, v)}
\]

- **application**: robustness of network \((\text{Scellato et al., 2011})\)

*drop in efficiency*: at time \(t = 150\), 20% of nodes are removed (sliding time window)
Burstiness

- defined for sequence of inter-event times $\tau$ (of single node or pair of nodes, or global)
- measures deviation from memoryless random Poisson process
- defined as

$$B(\tau) = \frac{\sigma_\tau - m_\tau}{\sigma_\tau + m_\tau} \in [-1, 1],$$

where $\sigma_\tau$ and $m_\tau$ denote the standard deviation and mean of the inter-contact times $\tau$

(Goh and Barabási, 2008)
Burstiness

- defined for sequence of inter-event times $\tau$ (of single node or pair of nodes, or global)
- measures deviation from memoryless random Poisson process
- defined as
  \[
  B(\tau) = \frac{\sigma_\tau - m_\tau}{\sigma_\tau + m_\tau} \in [-1, 1],
  \]
  where $\sigma_\tau$ and $m_\tau$ denote the standard deviation and mean of the inter-contact times $\tau$

- (a) $B(\tau) = 0 \Rightarrow$ a Poisson distribution
- (b) $B(\tau) = 1 \Rightarrow$ a maximally bursty sequence
- (c) $B(\tau) = -1 \Rightarrow$ a periodic sequence
Topological Overlap

.quantifies the persistency of edges through time (Tang et al., 2010b)

the topological overlap is defined as

\[
T_{to}(G) = \frac{1}{n} \sum_{u \in V} \frac{1}{T} \sum_{t=1}^{T-1} \frac{\sum_{v \in N(u)} \phi_{uv}^t \phi_{uv}^{t+1}}{\sqrt{\sum_{v \in N(u)} \phi_{uv}^t \sum_{v \in N(u)} \phi_{uv}^{t+1}}} \in [0, 1],
\]

where \(\phi_{uv}^t = 1\) iff. there exists a temporal edges between \(u\) and \(v\) at time \(t\) and zero otherwise.
Topological Overlap

▶ quantifies the persistency of edges through time
▶ the topological overlap is defined as

\[
T_{\text{to}}(G) = \frac{1}{n} \sum_{u \in V} \frac{1}{T} \sum_{t=1}^{T-1} \frac{\sum_{v \in N(u)} \phi_{uv}^t \phi_{uv}^{t+1}}{\sqrt{\sum_{v \in N(u)} \phi_{uv}^t} \sum_{v \in N(u)} \phi_{uv}^{t+1}} \in [0, 1],
\]

where \( \phi_{uv}^t = 1 \) iff. there exists a temporal edges between \( u \) and \( v \) at time \( t \) and zero otherwise

▶ value close to zero: many edges change between consecutive time steps

(Tang et al., 2010b)
Topological Overlap

- quantifies the persistency of edges through time  

- the topological overlap is defined as

\[
T_{to}(G) = \frac{1}{n} \sum_{u \in V} \frac{1}{T} \sum_{t=1}^{T-1} \frac{\sum_{v \in N(u)} \phi_{uv}^t \phi_{uv}^{t+1}}{\sqrt{\sum_{v \in N(u)} \phi_{uv}^t \sum_{v \in N(u)} \phi_{uv}^{t+1}}} \in [0, 1],
\]

where \( \phi_{uv}^t = 1 \) iff. there exists a temporal edges between \( u \) and \( v \) at time \( t \) and zero otherwise

- value close to zero: many edges change between consecutive time steps
- value close to one: means there are often only a few changes.
Temporal Clustering Coefficient

- the temporal clustering coefficient of node $u$ in time interval $I$ is defined as (Tang et al., 2009)

$$C_C(u, I) = \frac{\sum_{t \in I} \pi_t(u)}{|I| \binom{|N(u)|}{2}},$$

where $\pi_t(u) =$ number of edges between neighbors of $u$ at time $t$

- adaptation of static clustering coefficient
- quantifies how close a node’s neighbors are to forming a clique during time interval $I$
Temporal Clustering Coefficient

- Human contact network at MIT campus using bluetooth scanning every 5 minutes
- Global temporal clustering coefficient for each day
- Higher during middle of the week and no clustering on holidays

(Tang et al., 2009)
Centrality Measures – Finding Important Nodes
Centrality Measures

Task

- assign to each node \( v \in V \) a centrality value \( c(v) \)
- the higher \( c(v) \) the more important is \( v \)

- many centrality measures on static graphs:
  e.g., degree, closeness, betweenness, Katz centrality, PageRank, ...
Centrality Measures

**Task**

- assign to each node \( v \in V \) a centrality value \( c(v) \)
- the higher \( c(v) \) the more important is \( v \)

- many centrality measures on static graphs:
  - e.g., degree, closeness, betweenness, Katz centrality, PageRank, ...

Many important applications:

- identify key players, super spreaders, important persons, ...
- ranking web pages
- H-index used for ranking academics
Temporal Centrality Measures

- many common centrality measures are walk or path based
- classification in medial and radial

(Borgatti and Everett, 2006)
Temporal Centrality Measures

- many common centrality measures are walk or path based
- classification in medial and radial

**radial:** captures node influence over its neighbors
- count incoming or outgoing walks or paths

**medial:** captures node role as intermediary
- count walks or paths passing node

(Borgatti and Everett, 2006)
Temporal Centrality Measures

- many common centrality measures are walk or path based
- classification in medial and radial

**radial**: captures node influence over its neighbors
- count incoming or outgoing walks or paths

**medial**: captures node role as intermediary
- count walks or paths passing node

Common approach for temporal networks:
- replace path or walks with time-respecting paths or walks
## Temporal Centrality

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use-case</th>
</tr>
</thead>
<tbody>
<tr>
<td>temporal degree</td>
<td>radial</td>
<td>-</td>
</tr>
<tr>
<td>temporal closeness</td>
<td>radial</td>
<td>paths</td>
</tr>
<tr>
<td>temporal pagerank</td>
<td>radial</td>
<td>walks</td>
</tr>
<tr>
<td>temporal katz</td>
<td>radial</td>
<td>walks</td>
</tr>
<tr>
<td>temporal H-index</td>
<td>radial</td>
<td>walks</td>
</tr>
<tr>
<td>temporal betweenness</td>
<td>medial</td>
<td>paths</td>
</tr>
<tr>
<td>temporal walk centrality</td>
<td>medial</td>
<td>walks</td>
</tr>
</tbody>
</table>

- Choosing the right centrality measure depends on use-case
- Many further temporal centrality variants, e.g., temporal eigenvector, temporal gravity, etc.

(Hu et al., 2015; Rocha and Masuda, 2014; Tang et al., 2010a; Tsalouchidou et al., 2020; Bi et al., 2021; Elmezain et al., 2021; Zaoli et al., 2019; Tao et al., 2022; Taylor et al., 2021; Rozenshtein and Gionis, 2016) ...
Temporal Centrality

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use-case</th>
</tr>
</thead>
<tbody>
<tr>
<td>temporal degree</td>
<td>radial</td>
<td>identify nodes with high degree</td>
</tr>
<tr>
<td>temporal closeness</td>
<td>radial</td>
<td>paths identify nodes that can reach other nodes fast (or can be reached fast)</td>
</tr>
<tr>
<td>temporal pagerank</td>
<td>radial</td>
<td>walks adapts static pagerank to capture concept drift</td>
</tr>
<tr>
<td>temporal katz</td>
<td>radial</td>
<td>walks identify nodes with many incoming temporal walks</td>
</tr>
<tr>
<td>temporal H-index</td>
<td>radial</td>
<td>walks identification of super-spreaders</td>
</tr>
<tr>
<td>temporal betweenness</td>
<td>medial</td>
<td>paths identify nodes passed by many optimal temporal paths</td>
</tr>
<tr>
<td>temporal walk centrality</td>
<td>medial</td>
<td>walks identify nodes that can obtain and distribute information</td>
</tr>
</tbody>
</table>

Choosing the right centrality measure depends on use-case

Many further temporal centrality variants, e.g., temporal eigenvector, temporal gravity, etc. (Hu et al., 2015; Rocha and Masuda, 2014; Tang et al., 2010a; Tsalouchidou et al., 2020; Bi et al., 2021; Elmezain et al., 2021; Zaoli et al., 2019; Tao et al., 2022; Taylor et al., 2021; Rozenshtein and Gionis, 2016) ...
From Static to Temporal Closeness

Static harmonic closeness:

\[ C_c(u) = \sum_{v \in V \setminus \{u\}} \frac{1}{d_s(u, v)} \]

- \(d_s(u, v)\) is shortest path distance
- high centrality means short paths to many other nodes
- temporal: replace \(d_s(u, v)\) with temporal distance
- several different variants
  (Wu et al., 2014; Crescenzi et al., 2020; Tang et al., 2010a; Santoro et al., 2011; Gao et al., 2015)
Temporal Closeness

Harmonic temporal closeness for $u \in V$:

$$c(u) = \sum_{v \in V \setminus \{u\}} \frac{1}{d(u, v)}$$

$d(u, v)$ is the minimum duration distance (i.e., arrival time - starting time).

Use case

- find nodes that spread information fast

Computation:

- call minimum duration streaming algorithm (Wu et al., 2014) for each node
- lack of scalability

(Oettershagen and Mutzel, 2022)
Temporal Closeness – Top-k Computation

**Top-k closeness problem**: find all nodes with one of the $k$ topmost closeness values

**Top-k closeness computation**

- for each vertex $u \in V$
  - run min. duration algorithm to compute $d(u, v)$ for all $v \in V$
  - if upper bound of $c(u)$ is smaller than $k$-th largest value: stop computation early

(Oettershagen and Mutzel, 2022)
Temporal Closeness – Index

**Problem:** rank all nodes according to temporal closeness.

**Indexing approach**
- index to speed up minimum duration computation
- **two phases:** (i) indexing and (ii) query phase

(Oettershagen and Mutzel, 2023)
Temporal Closeness – Index

**Construction:**
- construct \( k \) subgraphs \( \{S_1, \ldots, S_k\} = S \)
- find mapping \( f : V \to S \) that assigns to each \( S_j \in S \) all vertices \( v \in V \) s.t. all edges reachable from \( v \) are in \( f(v) = S_j \)
- minimize size \( \max_{S \in S} \{|S|\} \)

Optimal assignment is NP-hard
Approximation ratio of greedy:

\[
\frac{\text{size(GREEDY)}}{\text{size(OPT)}} \leq \frac{k}{\delta},
\]

with \(1 \leq \delta \leq k\) depending on the topology of the graph.

**Time complexity:** \(O(nmk)\)

- \(n = |V|\) rounds, \(m = |E|\)
- each round determine \(S_j\) such that greedy choice is minimal in \(O(m)\) for \(j \in \{1, \ldots, k\}\)

**Shared memory parallelization:** \(O(\frac{nmk}{P})\) using \(P\) processors (CREW)
Temporal Closeness – Index

OOT—Out of time after 7 days. number of subgraphs $k = 2048$.

| Data set     | $|V|$ | $|E|$ | Baseline  | Top-100 | SubStream |
|--------------|-----|-----|----------|---------|-----------|
| Infectious   | 10 972 | 415 912 | 12.06 s | 2.25 s  | 1.51 s    |
| AskUbuntu    | 159 316 | 964 437 | 229.73 s | 132.53 s | 102.46 s  |
| Prosper      | 89 269  | 3 394 978 | 1 665.20 s | 260.87 s | 109.33 s  |
| Arxiv        | 28 093   | 4 596 803 | 630.60 s | 398.50 s | 286.86 s  |
| Youtube      | 3 223 585 | 9 375 374 | 145.98 h | 81.21 h  | 59.72 h   |
| StackOverflow| 2 464 606 | 17 823 525 | OOT      | 107.66 h | 86.49 h   |

- **Baseline**: Streaming algorithm (Wu et al., 2014)
- **Top-100**: Top-$k$ algorithm with $k = 100$
- **SubStream**: Index based computation
## Temporal Centrality

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use-case</th>
</tr>
</thead>
<tbody>
<tr>
<td>temporal degree</td>
<td>radial</td>
<td>identify nodes with high degree</td>
</tr>
<tr>
<td>temporal closeness</td>
<td>radial</td>
<td>identify nodes that can reach other nodes fast (or can be reached fast)</td>
</tr>
<tr>
<td>temporal pagerank</td>
<td>radial</td>
<td>adapts static pagerank to capture concept drift</td>
</tr>
<tr>
<td>temporal katz</td>
<td>radial</td>
<td>identify nodes with many incoming temporal walks</td>
</tr>
<tr>
<td>temporal H-index</td>
<td>radial</td>
<td>identification of super-spreaders</td>
</tr>
<tr>
<td>temporal betweenness</td>
<td>medial</td>
<td>identify nodes passed by many optimal temporal paths</td>
</tr>
<tr>
<td>temporal walk centrality</td>
<td>medial</td>
<td>identify nodes that can obtain and distribute information</td>
</tr>
</tbody>
</table>

- choosing the right centrality measure depends on use-case
- many further temporal centrality variants, e.g., temporal eigenvector, temporal gravity, etc.  
  (Hu et al., 2015; Rocha and Masuda, 2014; Tang et al., 2010a; Tsalouchidou et al., 2020; Bi et al., 2021; Elmezain et al., 2021; Zaoli et al., 2019; Tao et al., 2022; Taylor et al., 2021; Rozenshtein and Gionis, 2016) ...
Static H-Index

- The H-index was originally proposed by J. E. Hirsch 2005
  → measuring the productivity and impact of scientists

- The maximum value of $h$ such that the author has published at least $h$ papers that have each been cited at least $h$ times

<table>
<thead>
<tr>
<th>paper ID</th>
<th>citations</th>
<th>counted in H-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>No</td>
</tr>
</tbody>
</table>
Static H-Index

- The H-index was originally proposed by J. E. Hirsch 2005 → measuring the productivity and impact of scientists
- The maximum value of $h$ such that the author has published at least $h$ papers that have each been cited at least $h$ times

<table>
<thead>
<tr>
<th>paper ID</th>
<th>citations</th>
<th>counted in H-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>No</td>
</tr>
</tbody>
</table>

Recently used for quantifying spreading influence (Lü et al., 2016)
Static H-Index

\[ \mathcal{H} : \mathcal{M} \to \mathbb{N}_0 \] returns for a multiset of integers \( S \subseteq \{ s \mid s \in \mathbb{N}_0 \} \) the maximum integer \( i \) such that there are at least \( i \) elements \( s \) in \( S \) with \( s \geq i \)
Static H-Index

- \( \mathcal{H} : \mathcal{M} \rightarrow \mathbb{N}_0 \) returns for a multiset of integers \( S \subseteq \{ s \mid s \in \mathbb{N}_0 \} \) the maximum integer \( i \) such that there are at least \( i \) elements \( s \) in \( S \) with \( s \geq i \)

\( n \)-th order H-index \( s_u^{(n)} \) of a node \( u \) in a static graph:
- let \( s_u^{(0)} = \delta(u) \) the degree of node \( u \), then

\[
\begin{align*}
    s_u^{(n)} &= \mathcal{H} \left( \{ s_v^{(n-1)} \mid v \in V \text{ and } v \text{ is neighbor of } u \} \right) \\
(\text{L"u et al., 2016})
\end{align*}
\]
- the value of \( s_u^{(1)} \) corresponds to the H-index of \( u \)
n-th Order Temporal H-Index

- the multiset $\mathcal{N}(v, t)$ contains all pairs of nodes and times $(w, t_w)$ such that there is a temporal edge from $v$ to $w$ leaving at time $t' \geq t$ and arriving at time $t_w$
n-th Order Temporal H-Index

- the multiset $\mathcal{N}(v, t)$ contains all pairs of nodes and times $(w, t_w)$ such that there is a temporal edge from $v$ to $w$ leaving at time $t' \geq t$ and arriving at time $t_w$

**Definition**

The $n$-th order temporal H-index of a node $v \in V$ is defined as $h_v^{(n)} = h_v^{(n)}$ with

$$h_v^{(n)} = \mathcal{H} \left( \left\{ h_w^{(n-1)} \mid (w, t_w) \in \mathcal{N}(v, t) \right\} \right),$$

and $h_v^{(0)} = |\mathcal{N}(v, t)|$. 
n-th Order Temporal H-Index

- the multiset $\mathcal{N}(v, t)$ contains all pairs of nodes and times $(w, t_w)$ such that there is a temporal edge from $v$ to $w$ leaving at time $t' \geq t$ and arriving at time $t_w$

**Definition**

The $n$-th order temporal H-index of a node $v \in V$ is defined as $h_v^{(n)} = h_v^{(n)}$ with

$$h_v^{(n)} = \mathcal{H} \left( \left\{ h_w^{(n-1)} \middle| (w, t_w) \in \mathcal{N}(v, t) \right\} \right),$$

and $h_v^{(0)} = |\mathcal{N}(v, t)|$.

**Computation:**

- single-pass streaming algorithm for each node $i$-th order H-indices for $0 \leq i \leq n$
- running time in $\mathcal{O}(|E|n\delta_{\max})$ and space in $\mathcal{O}(|V|n\delta_{\max})$

(Oettershagen et al., 2023b)
n-th Order Temporal H-Index

(a) Temporal network $G$.

(b) The reachability tree $\Gamma(f)$ for vertex $f$ in the temporal network shown in (a).

$$h_{f,0}^{(1)} = \mathcal{H}(\{h_{d,2}^{(0)}, h_{e,2}^{(0)}, h_{h,2}^{(0)}, h_{g,2}^{(0)}\}) = \mathcal{H}(\{3, 2, 4, 3\}) = 3$$
n-th Order Temporal H-Index

(a) Temporal network $G$.

(b) The reachability tree $\Gamma(f)$ for vertex $f$ in the temporal network shown in (a).

$h_{f,0}^{(2)} = $
n-th Order Temporal H-Index

(a) Temporal network $G$.

(b) The reachability tree $\Gamma(f)$ for vertex $f$ in the temporal network shown in (a).

$$h_{f,0}^{(2)} = H(\{ h_{d,2}^{(1)}, h_{e,2}^{(1)}, h_{h,2}^{(1)}, h_{g,2}^{(1)} \})$$
n-th Order Temporal H-Index

(a) Temporal network $G$.

(b) The reachability tree $\Gamma(f)$ for vertex $f$ in the temporal network shown in (a).

\[
h_{f,0}^{(2)} = \mathcal{H}(\{h_{d,2}^{(1)}, h_{e,2}^{(1)}, h_{h,2}^{(1)}, h_{g,2}^{(1)}\})
\]

\[
= \mathcal{H}(\{\mathcal{H}(\{h_{g,5}^{(0)}, h_{e,3}^{(0)}, h_{a,6}^{(0)}\}), \mathcal{H}(\{h_{d,3}^{(0)}, h_{h,4}^{(0)}\}), \mathcal{H}(\{h_{e,4}^{(0)}, h_{i,6}^{(0)}, h_{j,6}^{(0)}, h_{g,5}^{(0)}\}), \mathcal{H}(\{h_{c,5}^{(0)}, h_{d,5}^{(0)}, h_{h,5}^{(0)}\})\})}
\]
n-th Order Temporal H-Index

(a) Temporal network \( \mathcal{G} \).

(b) The reachability tree \( \Gamma(f) \) for vertex \( f \) in the temporal network shown in (a).

\[
\begin{align*}
h^{(2)}_{f,0} &= \mathcal{H}(\{h^{(1)}_{d,2}, h^{(1)}_{e,2}, h^{(1)}_{h,2}, h^{(1)}_{g,2}\}) \\
&= \mathcal{H}(\mathcal{H}(\{h^{(0)}_{d,3}, h^{(0)}_{h,4}\}), \mathcal{H}(\{h^{(0)}_{e,4}, h^{(0)}_{i,6}, h^{(0)}_{j,6}, h^{(0)}_{g,5}\}), \mathcal{H}(\{h^{(0)}_{c,5}, h^{(0)}_{d,5}, h^{(0)}_{h,5}\})) \\
&= \mathcal{H}(\mathcal{H}(\{0, 1, 1\}), \mathcal{H}(\{2, 3\}), \mathcal{H}(\{0, 1, 1, 0\}), \mathcal{H}(\{1, 1, 2\}))
\end{align*}
\]
n-th Order Temporal H-Index

(a) Temporal network $G$.

(b) The reachability tree $\Gamma(f)$ for vertex $f$ in the temporal network shown in (a).

\[
h_{f,0}^{(2)} = \mathcal{H}(\{h_{d,2}^{(1)}, h_{e,2}^{(1)}, h_{h,2}^{(1)}, h_{g,2}^{(1)}\})
\]
\[
= \mathcal{H}(\mathcal{H}(\{h_{g,5}^{(0)}, h_{e,3}^{(0)}, h_{a,6}^{(0)}\}), \mathcal{H}(\{h_{d,3}^{(0)}, h_{h,4}^{(0)}\}), \mathcal{H}(\{h_{e,4}^{(0)}, h_{i,6}^{(0)}, h_{j,6}^{(0)}, h_{g,5}^{(0)}\})), \mathcal{H}(\{h_{c,5}^{(0)}, h_{d,5}^{(0)}, h_{h,5}^{(0)}\}))
\]
\[
= \mathcal{H}(\mathcal{H}(\{0, 1, 1\}), \mathcal{H}(\{2, 3\}), \mathcal{H}(\{0, 1, 1, 0\}), \mathcal{H}(\{1, 1, 2\})\}) = \mathcal{H}(\{1, 2, 1, 1\}) = 1
\]
n-th Order Temporal H-Index

Use Case: Influential spreader identification

▶ computed for different infection probabilities $\beta$ the mean node influence $R_u$ over 1000 independent SIR simulations leading to the SIR node rankings

▶ compared the SIR rankings with those obtained by the centrality measures using the Kendall $\tau_b$ rank correlation measure

![Graphs showing comparisons of different centrality measures for different networks](image)

(a) Malawi  
(b) FacebookMsg  
(c) Email
## Temporal Centrality

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use-case</th>
</tr>
</thead>
<tbody>
<tr>
<td>temporal degree</td>
<td>radial</td>
<td>identify nodes with high degree</td>
</tr>
<tr>
<td>temporal closeness</td>
<td>radial</td>
<td>identify nodes that can reach other nodes fast (or can be reached fast)</td>
</tr>
<tr>
<td>temporal pagerank</td>
<td>radial</td>
<td>walks adapts static pagerank to capture concept drift</td>
</tr>
<tr>
<td>temporal katz</td>
<td>radial</td>
<td>walks identify nodes with many incoming temporal walks</td>
</tr>
<tr>
<td>temporal H-index</td>
<td>radial</td>
<td>walks identification of super-spreaders</td>
</tr>
<tr>
<td>temporal betweenness</td>
<td>medial</td>
<td>paths identify nodes passed by many optimal temporal paths</td>
</tr>
<tr>
<td>temporal walk centrality</td>
<td>medial</td>
<td>walks identify nodes that can obtain and distribute information</td>
</tr>
</tbody>
</table>

- Choosing the right centrality measure depends on use-case
- Many further temporal centrality variants, e.g., temporal eigenvector, temporal gravity, etc. (Hu et al., 2015; Rocha and Masuda, 2014; Tang et al., 2010a; Tsalouchidou et al., 2020; Bi et al., 2021; Elmezain et al., 2021; Zaoli et al., 2019; Tao et al., 2022; Taylor et al., 2021; Rozenshtein and Gionis, 2016) ...
Betweenness Centrality

- node importance of \( u \in V \) in terms of number of optimal paths visiting \( u \)

\[
B(u) = \sum_{s \neq u \neq z \in V} \frac{\sigma_{s,z}(u)}{\sigma_{s,z}}
\]

- \( \sigma_{s,z} \) number of shortest \( s, z \)-paths
- \( \sigma_{s,z}(u) \) number of shortest \( s, z \)-paths visiting \( u \)
Betweenness Centrality

- node importance of $u \in V$ in terms of number of optimal paths visiting $u$

$$B(u) = \sum_{s \neq u \neq z \in V} \frac{\sigma_{s,z}(u)}{\sigma_{s,z}}$$

- $\sigma_{s,z}$ number of shortest $s, z$-paths
- $\sigma_{s,z}(u)$ number of shortest $s, z$-paths visiting $u$

- Brandes' algorithm: iteratively calculates the shortest paths using dynamic programming, efficiently updating centrality scores
Betweenness Centrality

- node importance of $u \in V$ in terms of number of optimal paths visiting $u$

\[ B(u) = \sum_{s \neq u \neq z \in V} \frac{\sigma_{s,z}(u)}{\sigma_{s,z}} \]

- $\sigma_{s,z}$ number of shortest $s, z$-paths
- $\sigma_{s,z}(u)$ number of shortest $s, z$-paths visiting $u$

- **Brandes' algorithm**: iteratively calculates the shortest paths using dynamic programming, efficiently updating centrality scores

- **idea**: replace shortest paths with optimal temporal paths \(\text{(Kim and Anderson, 2012)}\)
Betweenness Centrality

- node importance of $u \in V$ in terms of number of optimal paths visiting $u$

$$B(u) = \sum_{s \neq u \neq z \in V} \frac{\sigma_{s,z}(u)}{\sigma_{s,z}}$$

- $\sigma_{s,z}$ number of shortest $s, z$-paths
- $\sigma_{s,z}(u)$ number of shortest $s, z$-paths visiting $u$

- **Brandes' algorithm**: iteratively calculates the shortest paths using dynamic programming, efficiently updating centrality scores

- **idea**: replace shortest paths with optimal temporal paths (Kim and Anderson, 2012)

- computing betweenness values is at least as hard as counting optimal paths
### Temporal Betweenness

Overview over the complexity of computing temporal betweenness centrality

<table>
<thead>
<tr>
<th>path type</th>
<th>strict</th>
<th>non-strict</th>
</tr>
</thead>
<tbody>
<tr>
<td>min.-hop</td>
<td>$O(n^3 \cdot T^2)$</td>
<td>$O(n^3 \cdot T^2)$</td>
</tr>
<tr>
<td>earliest arrival</td>
<td>#P-hard</td>
<td>#P-hard</td>
</tr>
<tr>
<td>fastest</td>
<td>#P-hard</td>
<td>#P-hard</td>
</tr>
<tr>
<td>prefix-earliest arrival</td>
<td>$O(n \cdot m \cdot \log m)$</td>
<td>#P-hard</td>
</tr>
<tr>
<td>min.-hop earliest arrival</td>
<td>$O(n^3 \cdot T^2)$</td>
<td>$O(n^3 \cdot T^2)$</td>
</tr>
</tbody>
</table>

**Prefix-earliest arrival:** every prefix of the temporal path is an earliest arrival path

▶ computation using adapted Brandes’ algorithm  

(Buß et al., 2020)
Temporal Betweenness

- more possible temporal walks and path types
  - characterization of properties such that paths are efficient countable

(Rymar et al., 2021)
Temporal Betweenness

- more possible temporal walks and path types (Rymar et al., 2021)
  - characterization of properties such that paths are efficient countable

- approximation algorithms (Santoro and Sarpe, 2022; Cruciani, 2023)
  - sampling-based approximations for different kinds of temporal path types
Temporal Betweenness

- more possible temporal walks and path types
  - characterization of properties such that path are efficient countable

- approximation algorithms
  - sampling-based approximations for different kinds of temporal path types

- comparison of different proxies for temporal betweenness
  - replacing global centrality with local pass-through-degree
## Temporal Centrality

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use-case</th>
</tr>
</thead>
<tbody>
<tr>
<td>temporal degree</td>
<td>radial</td>
<td>identify nodes with high degree</td>
</tr>
<tr>
<td>temporal closeness</td>
<td>radial</td>
<td>identify nodes that can reach other nodes fast (or can be reached fast)</td>
</tr>
<tr>
<td>temporal pagerank</td>
<td>radial</td>
<td>adapts static pagerank to capture concept drift</td>
</tr>
<tr>
<td>temporal katz</td>
<td>radial</td>
<td>identify nodes with many incoming temporal walks</td>
</tr>
<tr>
<td>temporal H-index</td>
<td>radial</td>
<td>identification of super-spreaders</td>
</tr>
<tr>
<td>temporal betweenness</td>
<td>medial</td>
<td>identify nodes passed by many optimal temporal paths</td>
</tr>
<tr>
<td>temporal walk centrality</td>
<td>medial</td>
<td>identify nodes that can obtain and distribute information</td>
</tr>
</tbody>
</table>

- Choosing the right centrality measure depends on use-case
- Many further temporal centrality variants, e.g., temporal eigenvector, temporal gravity, etc.  
  (Hu et al., 2015; Rocha and Masuda, 2014; Tang et al., 2010a; Tsalouchidou et al., 2020; Bi et al., 2021; Elmezain et al., 2021; Zaoli et al., 2019; Tao et al., 2022; Taylor et al., 2021; Rozenshtein and Gionis, 2016) ...
Temporal Walk Centrality

▶ let $Y_{in}(u, t)$ and $Y_{out}(u, t)$ be the sets of incoming and outgoing temporal walks, resp., at node $u$ and time $t$. 

Temporal Walk Centrality

The temporal walk centrality of a vertex $u \in V$ is

$$C(u) = \sum_{t_1, t_2 \in T(G), t_1 \leq t_2} W_{in}(u, t_1) \cdot W_{out}(u, t_2) \cdot \Phi_m(t_1, t_2).$$

Captures a node's ability to obtain and pass on information.
Temporal Walk Centrality

- let $Y_{in}(u, t)$ and $Y_{out}(u, t)$ be the sets of incoming and outgoing temporal walks, resp., at node $u$ and time $t$.
- for weighting functions $\tau_{\Phi_{in}}$ and $\tau_{\Phi_{out}}$, define

$$W_{in}(u, t) = \sum_{\omega \in Y_{in}(u, t)} \tau_{\Phi_{in}}(\omega) \quad \text{and} \quad W_{out}(u, t) = \sum_{\omega \in Y_{out}(u, t)} \tau_{\Phi_{out}}(\omega)$$
Temporal Walk Centrality

- Let $Y_{in}(u, t)$ and $Y_{out}(u, t)$ be the sets of incoming and outgoing temporal walks, resp., at node $u$ and time $t$.
- For weighting functions $\tau_{\Phi_{in}}$ and $\tau_{\Phi_{out}}$, define

$$W_{in}(u, t) = \sum_{\omega \in Y_{in}(u,t)} \tau_{\Phi_{in}}(\omega) \quad \text{and} \quad W_{out}(u, t) = \sum_{\omega \in Y_{out}(u,t)} \tau_{\Phi_{out}}(\omega)$$

Temporal Walk Centrality

The temporal walk centrality of a vertex $u \in V$ is

$$C(u) = \sum_{t_1, t_2 \in T(G), t_1 \leq t_2} (W_{in}(u, t_1) \cdot W_{out}(u, t_2) \cdot \Phi_m(t_1, t_2)).$$
Temporal Walk Centrality

- let \( Y_{in}(u, t) \) and \( Y_{out}(u, t) \) be the sets of incoming and outgoing temporal walks, resp., at node \( u \) and time \( t \).
- for weighting functions \( \tau_{\Phi_{in}} \) and \( \tau_{\Phi_{out}} \), define

\[
W_{in}(u, t) = \sum_{\omega \in Y_{in}(u, t)} \tau_{\Phi_{in}}(\omega) \quad \text{and} \quad W_{out}(u, t) = \sum_{\omega \in Y_{out}(u, t)} \tau_{\Phi_{out}}(\omega)
\]

The temporal walk centrality of a vertex \( u \in V \) is

\[
C(u) = \sum_{t_1, t_2 \in T(G), t_1 \leq t_2} \left( W_{in}(u, t_1) \cdot W_{out}(u, t_2) \cdot \Phi_m(t_1, t_2) \right).
\]

Captures a node's ability to obtain and pass on information
Temporal Walk Centrality

Directed line graph

- temporal walk in $G$ of length $\ell + 1 \iff$ walk of length $\ell$ in $D(G)$
Temporal Walk Centrality

Directed line graph

- temporal walk in $G$ of length $\ell + 1 \iff$ walk of length $\ell$ in $D(G)$
- walks can be computed by matrix powers: Neumann series and the identity $\sum_{\ell=0}^{\infty} A^\ell = (I - A)^{-1}$ holds if the sum converges—guaranteed when largest absolute eigenvalue less than one
Temporal Walk Centrality

▶ temporal walk in $G$ of length $\ell + 1 \Leftrightarrow$ walk of length $\ell$ in $D(G)$
▶ walks can be computed by matrix powers: Neumann series and the identity $\sum_{\ell=0}^{\infty} A^\ell = (I - A)^{-1}$ holds if the sum converges—guaranteed when largest absolute eigenvalue less than one
▶ computation in $\mathcal{O}(|E|^{2.373})$ using matrix inversion
▶ approximation with power iteration in $\mathcal{O}(k(|E|^2))$
Temporal Walk Centrality

**Two-Pass Streaming Algorithm**

- **input:** edge sequence in chronological order, ties broken arbitrarily
- **forward pass** for computing incoming walks for $W_{in}$
- **backward pass** for computing outgoing walks for $W_{out}$

- running time $O(|E| \cdot \tau_{max})$
- $\tau_{max}$ the largest cardinality of availability or arrival times at a node
Temporal Walk Centrality

(a) temporal walk centr.
(b) temporal between.
(c) static walk between.

- *enron* email subgraph: of 38 nodes and 541 temporal edges.
- colors represent centrality value: darker $\rightarrow$ higher centrality.
Temporal Centrality

- **Temporal closeness**
  - many different variants
  - intuitive, several approaches for improving computation times
Temporal Centrality

- **Temporal closeness**
  - many different variants
  - intuitive, several approaches for improving computation times

- **Temporal H-index**
  - non-intuitive definition
  - able to capture the spreading capabilities well, efficient

Choosing the right centrality measure depends on data and use-case.
Temporal Centrality

- **Temporal closeness**
  - many different variants
  - intuitive, several approaches for improving computation times

- **Temporal H-index**
  - non-intuitive definition
  - able to capture the spreading capabilities well, efficient

- **Temporal betweenness**
  - not scalable and many variants hard to compute
  - intuitive, several approaches for improving computation times
Temporal Centrality

- **Temporal closeness**
  - many different variants
  - intuitive, several approaches for improving computation times

- **Temporal H-index**
  - non-intuitive definition
  - able to capture the spreading capabilities well, efficient

- **Temporal betweenness**
  - not scalable and many variants hard to compute
  - intuitive, several approaches for improving computation times

- **Temporal walk centrality**
  - no ranking of sinks or sources
  - intuitive and efficient

Choosing the right centrality measure depends on data and use-case.
Temporal Centrality

- **Temporal closeness**
  - many different variants
  - intuitive, several approaches for improving computation times

- **Temporal H-index**
  - non-intuitive definition
  - able to capture the spreading capabilities well, efficient

- **Temporal betweenness**
  - not scalable and many variants hard to compute
  - intuitive, several approaches for improving computation times

- **Temporal walk centrality**
  - no ranking of sinks or sources
  - intuitive and efficient

- choosing the right centrality measure depends on data and use-case
Temporal Core Decompositions
k-Core Decomposition

- \( k \)-core is a max. subgraph \( G_k \) of \( G \), s.t. every node in \( G_k \) has at least \( k \) neighbors in \( G_k \)
- node \( u \) has core number \( c(u) = k \) if \( u \) belongs to a \( k \)-core but not the \( k + 1 \)-core

(Seidman, 1983; Kong et al., 2019)
Applications

- identifying communities and dense graphs in social networks

(Malliaros et al., 2020; Kong et al., 2019)
Applications

- identifying communities and dense graphs in social networks
- anomaly detection

(Malliaros et al., 2020; Kong et al., 2019)
Applications

- identifying communities and dense graphs in social networks
- anomaly detection
- network visualization

(Malliaros et al., 2020; Kong et al., 2019)
Applications

- identifying communities and dense graphs in social networks
- anomaly detection
- network visualization
- analyzing backbone structure of the internet

(Malliaros et al., 2020; Kong et al., 2019)
Applications

- identifying communities and dense graphs in social networks
- anomaly detection
- network visualization
- analyzing backbone structure of the internet
- network resilience and robustness

(Malliaros et al., 2020; Kong et al., 2019)
Applications

- identifying communities and dense graphs in social networks
- anomaly detection
- network visualization
- analyzing backbone structure of the internet
- network resilience and robustness
- study of spreading processes

(Malliaros et al., 2020; Kong et al., 2019)
Applications

- identifying communities and dense graphs in social networks
- anomaly detection
- network visualization
- analyzing backbone structure of the internet
- network resilience and robustness
- study of spreading processes
- analyzing functional brain networks

(Malliaros et al., 2020; Kong et al., 2019)
## Temporal k-Core Decompositions

<table>
<thead>
<tr>
<th>Variant</th>
<th>Ref.</th>
<th>Running Time</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Historical $k$-core</td>
<td>(Yu et al., 2021)</td>
<td>$\mathcal{O}(\log m + m_I)$</td>
<td>static cores spanning fixed interval $I$</td>
</tr>
<tr>
<td>Time-range $k$-core</td>
<td>(Yang et al., 2023)</td>
<td>$\mathcal{O}(\log m +</td>
<td>I</td>
</tr>
<tr>
<td>$(k, h)$-core</td>
<td>(Wu et al., 2015)</td>
<td>$\mathcal{O}(n + m)$</td>
<td>parallel temporal edges</td>
</tr>
<tr>
<td>Span-core</td>
<td>(Galimberti et al., 2020)</td>
<td>$\mathcal{O}(</td>
<td>T</td>
</tr>
<tr>
<td>$(k, \Delta)$-core</td>
<td>(Oettershagen et al., 2023a)</td>
<td>$\mathcal{O}(m \cdot \delta_m)$</td>
<td>based on temporal edge degree</td>
</tr>
</tbody>
</table>

- more variants available (Lotito and Montresor, 2020; Hung and Tseng, 2021; Qin et al., 2022, 2020; Li et al., 2018; Oettershagen et al., 2023b) ...

- choosing the right one depends on available data and application
Historical and Time-Range $k$-Cores

- definitions based on time window graph
  - static, aggregated graph for time window $I$
- historical: find a static at least $k$-core in time window graph for given time interval $I$
- time-range: find all distinct at least $k$-cores in all possible time windows in time interval $I$

(Yu et al., 2021; Yang et al., 2023)
Historical and Time-Range $k$-Cores

6-cores of Prof. Jiawei Han’s ego network on the DBLP snapshots

- **straight-forward computation**: restrict to interval (or subintervals)
- **index-based solution** to support the $k$-core query for every possible time window and integer $k$

(Yu et al., 2021; Yang et al., 2023)
(k, h)-Cores

**Definition**

(k, h)-core is the largest subgraph H such that every v in H must have at least k neighbors in H, where each such neighbor must be connected to v with at least h temporal edges.

- can be interpreted as core decomposition for multi(-layer) graphs

(a) Temporal graph G

(b) Underlying multi graph

- nodes \{a, b, c, d\} induce a (2, 2)-core

(Wu et al., 2015)
Span-Cores

**Definition**

- the \((k, \Delta)\)-core is a maximal set of vertices \(C\) such that \(C\) is a \(k\)-core over the complete span of time interval \(\Delta\) (each edge of the core exists in each time step in \(\Delta\))
- a span-core is maximal if no other span-core dominates it in \(k\) or \(\Delta\)

Applications:
- community search
- identify temporal patterns
- anomaly detection
- graph embedding and vertex classification
- containing or maximizing spreading (Galimberti et al., 2020; Ciaperoni et al., 2020)
Span-Cores

Definition

- the \((k, \Delta)\)-core is a maximal set of vertices \(C\) such that \(C\) is a \(k\)-core over the complete span of time interval \(\Delta\) (each edge of the core exists in each time step in \(\Delta\))
- a span-core is maximal if no other span-core dominates it in \(k\) or \(\Delta\)

Applications:

- community search

(Galimberti et al., 2020; Ciaperoni et al., 2020)
Span-Cores

**Definition**

- the \((k, \Delta)\)-core is a maximal set of vertices \(C\) such that \(C\) is a \(k\)-core over the complete span of time interval \(\Delta\) (each edge of the core exists in each time step in \(\Delta\))
- a span-core is maximal if no other span-core dominates it in \(k\) or \(\Delta\)

**Applications:**

- community search
- identify temporal patterns

(Galimberti et al., 2020; Ciaperoni et al., 2020)
Span-Cores

**Definition**
- the \((k, \Delta)\)-core is a maximal set of vertices \(C\) such that \(C\) is a \(k\)-core over the complete span of time interval \(\Delta\) (each edge of the core exists in each time step in \(\Delta\))
- a span-core is maximal if no other span-core dominates it in \(k\) or \(\Delta\)

**Applications:**
- community search
- identify temporal patterns
- anomaly detection

(Galimberti et al., 2020; Ciaperoni et al., 2020)
Span-Cores

**Definition**

- the \((k, \Delta)\)-core is a maximal set of vertices \(C\) such that \(C\) is a \(k\)-core over the complete span of time interval \(\Delta\) (each edge of the core exists in each time step in \(\Delta\)).
- a span-core is maximal if no other span-core dominates it in \(k\) or \(\Delta\).

**Applications:**

- community search
- identify temporal patterns
- anomaly detection
- graph embedding and vertex classification

(Galimberti et al., 2020; Ciaperoni et al., 2020)
Span-Cores

**Definition**

- the \((k, \Delta)\)-core is a maximal set of vertices \(C\) such that \(C\) is a \(k\)-core over the complete span of time interval \(\Delta\) (each edge of the core exists in each time step in \(\Delta\))
- a span-core is maximal if no other span-core dominates it in \(k\) or \(\Delta\)

**Applications:**

- community search
- identify temporal patterns
- anomaly detection
- graph embedding and vertex classification
- containing or maximizing spreading

(Galimberti et al., 2020; Ciaperoni et al., 2020)
Span-Cores

**Application:** Temporal pattern identification

- temporal activity of a high school day

(Galimberti et al., 2020)
### Span-Cores

**Application:** Temporal pattern identification
- temporal activity of a high school day
- span-core decomposition detects time-evolving cohesive substructures

(Galimberti et al., 2020)
Span-Cores

**Application:** Temporal pattern identification

- temporal activity of a high school day
- span-core decomposition detects time-evolving cohesive substructures
- these completely disappear in the reshuffled data set

(Galimberti et al., 2020)
\((k, \Delta)\)-cores

**Definition**

- \(\Delta\)-degree of an edge is the minimum number of edges incident to one of its endpoints that have a temporal distance of at most \(\Delta\).
- The \((k, \Delta)\)-core is the inclusion-maximal edge-induced subgraph \(C_{(k, \Delta)}\) of \(G\) such that each temporal edge \(e = (\{u, v\}, t)\) in \(C_{(k, \Delta)}\) has at least a \(\Delta\)-degree of \(d_{\Delta}(e) \geq k + 1\).
Definition

- $\Delta$-degree of an edge is the minimum number of edges incident to one of its endpoints that have a temporal distance of at most $\Delta$.
- the $(k, \Delta)$-core is the inclusion-maximal edge-induced subgraph $C_{(k,\Delta)}$ of $G$ such that each temporal edge $e = (\{u, v\}, t)$ in $C_{(k,\Delta)}$ has at least a $\Delta$-degree of $d_{\Delta}(e) \geq k + 1$.

Each edge in a $(k, \Delta)$-core is at both ends incident to at least $k + 1$ edges in the core with temporal distance at most $\Delta$.

(Oettershagen et al., 2023a)
\((k, \Delta)\)-cores

**Application:**
- analyzing malicious retweets in the Twitter network
- the most inner cores only contain misinformation for \(\Delta = 1\) hour
Temporal k-Core Decompositions

- **Historical and time range** $k$-core
  - based on time window graphs but efficient
Temporal k-Core Decompositions

- **Historical and time range $k$-core**
  - based on time window graphs but efficient

- **$(k, h)$-core**
  - considers only multilayer graph, also efficient
Temporal k-Core Decompositions

- **Historical and time range k-core**
  - based on time window graphs but efficient

- **(k, h)-core**
  - considers only multilayer graph, also efficient

- **Span-core**
  - needs dense sequence of static graphs, also not very scalable
  - shown to be useful in wide range of applications
Temporal k-Core Decompositions

- **Historical and time range** $k$-core
  - based on time window graphs but efficient

- **$(k, h)$-core**
  - considers only multilayer graph, also efficient

- **Span-core**
  - needs dense sequence of static graphs, also not very scalable
  - shown to be useful in wide range of applications

- **$(k, \Delta)$-core**
  - can handle fine-grained data and efficient
Temporal k-Core Decompositions

- **Historical and time range $k$-core**
  - based on *time window graphs* but *efficient*

- **$(k, h)$-core**
  - considers only *multilayer graph*, also *efficient*

- **Span-core**
  - needs dense *sequence of static graphs*, also *not very scalable*
  - shown to be *useful in wide range of applications*

- **$(k, \Delta)$-core**
  - can handle *fine-grained data* and *efficient*

- choosing the right one depends on available data and application
Agenda

Part I: Introduction and Motivation
- models of temporal networks
- algorithmic approaches

Part II: Mining Temporal Networks A:
- connectivity, temporal properties
- centrality, cores

Part III: Mining Temporal Networks B:
- communities, patterns and events
- diffusion and random networks

Part IV: Tools and Code Libraries

Part V: Challenges, Open Problems, and Trends
Part IV

Mining Temporal Networks B
Community Detection
Temporal Communities

Identifying communities is a fundamental task in computer and network science.
Question: How do we define a temporal community?
Question: How do we define a temporal community?
Communities in Static Networks

“community” = “umbrella term”

- extensive surveys (Fortunato and Hric, 2016; Su et al., 2024)

- possible definitions
  - a community is a set of nodes, closer to each other than to the rest of the network
  - a community is a “dense” subgraph
Communities in Static Networks

“community” = “umbrella term”

- extensive surveys (Fortunato and Hric, 2016; Su et al., 2024)
- possible definitions
  - a community is a set of nodes, closer to each other than to the rest of the network
  - a community is a “dense” subgraph

- typical problem settings
  - a single community vs. network partitioning
  - overlapping vs. non-overlapping communities
  - local to some nodes vs. global
Community Detection in Static Network

Usual workflow (data analysis)

1. pick a problem setting (e.g., partition in \( k \) communities vs identify a single local community)

2. pick a proper metric to quantify the “density” of the community \( S \)
   - average degree: \( \frac{|E(S)|}{2|S|} \)
   - density: \( \frac{2|E(S)|}{|S|(|S|-1)} \)
   - conductance: \( \frac{\text{cut}(S,\bar{S})}{\min\{\text{vol}(S),\text{vol}(\bar{S})\}} \)
   - modularity
   - ...

3. identify/design proper algorithms to solve the problem

4. analyze data and, if needed repeat from steps 1. or 2.
Analyses and Applications

- **social networks**
  - link prediction, targeted advertisement, content moderation, etc.
Analyses and Applications

- **social networks**
  - link prediction, targeted advertisement, content moderation, etc..

- **financial networks**
  - fraud detection, money-laundering activities, etc..
Analyses and Applications

- **social networks**
  - link prediction, targeted advertisement, content moderation, etc..

- **financial networks**
  - fraud detection, money-laundering activities, etc..

- **collaboration networks**
  - identifying trending topics, important group of nodes, etc..

- ...
So what is new about temporal communities?
Temporal Evolution of Communities

Temporal networks allow us to study communities according to their \textit{temporal evolution}!

Some representative behaviors,

Growing community $C$

\begin{figure}
\centering
\includegraphics[width=\textwidth]{community_growth}
\caption{Illustration of growing community $C$ over time $T$.}
\end{figure}
Temporal Evolution of Communities

Temporal networks allow us to study communities according to their temporal evolution!

Some representative behaviors,

Growing community $C$

Shrinking community $C$
Temporal Evolution of Communities

Temporal networks allow us to study communities according to their temporal evolution!

Some representative behaviors,

- Growing community $C$
- Shrinking community $C$
- Periodic community $C$
Temporal Evolution of Communities

temporal networks allow us to study communities according to their **temporal evolution**!

Some representative behaviors,

- Growing community \( C \)
- Shrinking community \( C \)
- Periodic community \( C \)
- Bursty community \( C \)
Temporal Evolution of Communities

Temporal networks allow us to study communities according to their **temporal evolution**!

Some representative behaviors,

- Growing community \( C \)
- Shrinking community \( C \)
- Periodic community \( C \)
- Bursty community \( C \)
- Merging communities

Are there proposed taxonomies?
Community Detection in Temporal Networks

**Question** How many taxonomies exist?
Community Detection in Temporal Networks

- proposed taxonomies
  - (Aynaud et al., 2013)
  - (Aggarwal and Subbian, 2014)
  - (Enugala et al., 2015)
  - (Renaud and Naoki, 2016)
  - (Hartmann et al., 2016)
  - (Rossetti and Cazabet, 2018)
  - (Dakiche et al., 2019)
  - (Christopoulos and Tsichlas, 2022)
  - ...
Community Detection in Temporal Networks

- proposed taxonomies
  - (Aynaud et al., 2013)
  - (Aggarwal and Subbian, 2014)
  - (Enugala et al., 2015)
  - (Renaud and Naoki, 2016)
  - (Hartmann et al., 2016)
  - (Rossetti and Cazabet, 2018)
  - (Dakiche et al., 2019)
  - (Christopoulos and Tsichlas, 2022)
  - ...

No need to panic: recall our steps for community identification!
Temporal Community Detection

Proposed workflow (exploratory analysis), everything starts from data!

1. identify if temporal data is fine-grained or course-grained (!)
2. pick a problem setting (✓)
3. pick a proper metric to quantify the “temporal density” of the community $S$, encoding the desired temporal properties (!)
4. identify/design proper algorithms to solve the problem
5. analyze data and, if needed repeat from steps 1. or 2.

Lets make it more concrete!
BFF: Finding Lasting Communities (Semertzidis et al., 2016)

Data. Consider a temporal network $G = \{G_1, \ldots, G_T\}$

Setting. We want to find a single global dense community across all snapshots!

Implicit assumption. time is sufficiently course-grained so each snapshot has enough information

Metrics and temporal properties. Fix a time $t$, given $S \subseteq V$ we define,

$\triangleright d_{avg}(S, G_t) = \frac{1}{|S|} \sum_{u \in S} d(u, G_t[S]) = \frac{2|E(S, G_t)|}{|S|}$, and

$\triangleright d_{min}(S, G_t) = \min_{u \in S} d(u, G_t[S])$
BFF: Finding Lasting Communities (Semertzidis et al., 2016)

**Data.** Consider a temporal network $G = \{G_1, \ldots, G_T\}$

**Setting.** We want to find a single global dense community across all snapshots!

**Implicit assumption.** time is sufficiently course-grained so each snapshot has enough information

**Metrics and temporal properties.** Fix a time $t$, given $S \subseteq V$ we define,

- $d_{avg}(S, G_t) = \frac{1}{|S|} \sum_{u \in S} d(u, G_t[S]) = \frac{2|E(S, G_t)|}{|S|}$, and

- $d_{min}(S, G_t) = \min_{u \in S} d(u, G_t[S])$

Combining such values across snapshots, let $- \in \{avg, min\}$

- $g_{min}(d_-(S, G)) = \min_{t=1,\ldots,T} d_-(S, G_t)$

- $g_{avg}(d_-(S, G)) = \frac{1}{T} \sum_t d_-(S, G_t)$
BFF: Finding Lasting Communities (Semertzidis et al., 2016)

Data. Consider a temporal network $G = \{G_1, \ldots, G_T\}$

Setting. We want to find a single global dense community across all snapshots!

Implicit assumption. time is sufficiently course-grained so each snapshot has enough information

Metrics and temporal properties. Fix a time $t$, given $S \subseteq V$ we define,

$\left\{\begin{array}{ll}
\quad d_{\text{avg}}(S, G_t) &= \frac{1}{|S|} \sum_{u \in S} d(u, G_t[S]) = \frac{2|E(S, G_t)|}{|S|}, \\
\quad d_{\text{min}}(S, G_t) &= \min_{u \in S} d(u, G_t[S])
\end{array}\right.$

Combining such values across snapshots, let $- \in \{\text{avg}, \text{min}\}$

$\left\{\begin{array}{ll}
\quad g_{\text{min}}(d_{-}(S, G)) &= \min_{t=1, \ldots, T} d_{-}(S, G_t) \\
\quad g_{\text{avg}}(d_{-}(S, G)) &= \frac{1}{T} \sum_{t} d_{-}(S, G_t)
\end{array}\right.$

So we finally have a score for a community $S \subseteq V$, that is given $+, - \in \{\text{avg}, \text{min}\}$ we let

$f_{+, -}(S, G) = g_{+}(d_{-}(S, G))$
Given $G = (G_1, \ldots, G_T)$, let $+, - \in \{\text{avg}, \text{min}\}$ and a target density $f_{+, -}$ find a subset of vertices $S^* \subseteq V$ maximizing the objective $f_{+, -}(S, G)$ over all communities.

Inspecting the objective values
BFF: A Closer Look on the Metrics

**Problem**

Given $G = (G_1, \ldots, G_T)$, let $+, - \in \{\text{avg, min}\}$ and a target density $f_{+, -}$ find a subset of vertices $S^* \subseteq V$ maximizing the objective $f_{+, -}(S, G)$ over all communities.

Inspecting the objective values

- $f_{\text{min}, \text{min}}(S, G) = g_{\text{min}}(d_{\text{min}}(S, G)) = \min_{t=1, \ldots, T} \min_{u \in S} d(u, G_t[S])$, minimum degree on every snapshot of all vertices $v \in S^*$ is high

- $f_{\text{min}, \text{avg}}(S, G) = g_{\text{min}}(d_{\text{avg}}(S, G)) = \min_{t=1, \ldots, T} \frac{2|E(S, G_t)|}{|S|}$, average degree of each node in $S$ is large on each snapshot

- $f_{\text{avg}, \text{min}}(S, G) = g_{\text{avg}}(d_{\text{min}}(S, G)) = \frac{1}{T} \sum_t \min_{u \in S} d(u, G_t[S])$, on average the minimum degree of each node in $S$ is high (more flexible than $f_{\text{min}, \text{min}}$)

- $f_{\text{avg}, \text{avg}}(S, G) = g_{\text{avg}}(d_{\text{avg}}(S, G)) = \frac{1}{T} \sum_t \frac{2|E(S, G_t)|}{|S|}$, on average the average degree of each node in $S$ is high
BFF: Algorithm

Computing the solution, FindBFF (Inspired by Charikar (2000))

1. iteratively and greedily peel $V$ removing at each step $v = \arg \min_{v \in V} \text{score}(v, G[V])$
2. compute the density target density on the remaining network where $V = V \setminus \{v\}$
3. return the vertex-set $S \subseteq V$ maximizing the target density over all $O(n)$ iterations

How is $\text{score}(v, G[V])$ computed? Depends on the objective $f$

- For $f_{\min}$, $\min_{S \subseteq V} \text{score}(S, G)$ we set $\text{score}(v, G[V]) = \min_{t=1, \ldots, T} \text{density}(v, G_t[V])$
- For $f_{\text{avg}}$, $\text{avg}_{S \subseteq V} \text{score}(S, G)$ we set $\text{score}(v, G[V]) = \frac{1}{T} \sum_{t=1}^{T} \text{density}(v, G_t[V])$

Resulting algorithms run in $O(nT + P T |E_t|)$

- $\text{FindBFF-MM (}f_{\min}$), finds the optimal solution
- $\text{FindBFF-AA (}f_{\text{avg}}$, finds a $\frac{1}{2}$-approximation
- $\text{BFF-AM (}f_{\text{avg}}, f_{\min})$ is NP-hard, FindBFF achieves an approximation at most $O(n)$
- $\text{BFF-MA (}f_{\min}, f_{\text{avg}}$), “complexity = ?”, FindBFF achieves an approximation at most $O(n - 1/2)$
BFF: Algorithm

Computing the solution, FindBFF (Inspired by Charikar (2000))

1. iteratively and greedily peel $V$ removing at each step $v = \arg\min_{v \in V} score(v, G[V])$
2. compute the density target density on the remaining network where $V = V \setminus \{v\}$
3. return the vertex-set $S \subseteq V$ maximizing the target density over all $O(n)$ iterations

How is $score(v, G[V])$ computed? Depends on the objective $f$

- for $f_{\text{min, min}}(S, G)$ we set $score(v, G[V_i]) = \min_{t=1,\ldots,T} d(v, G_t[V])$
- for $f_{\text{avg, avg}}(S, G)$ we set $score(v, G[V_i]) = \frac{1}{T} \sum_{t} d(v, G_t[V])$

Resulting algorithms run in $O(nT + P_t|E_t)$

- FindBFF-MM ($f_{\text{min}}$), finds the optimal solution
- FindBFF-AA ($f_{\text{avg}}$), finds a $\frac{1}{2}$-approximation
- BFF-AM ($f_{\text{avg}}, f_{\text{min}}$) is $\text{NP}$-hard, FindBFF achieves an approximation at most $O(n - 1)$
- BFF-MA ($f_{\text{min}}, f_{\text{avg}}$), “complexity = ?”, FindBFF achieves an approximation at most $O(n - 1/2)$
BFF: Algorithm

Computing the solution, FindBFF (Inspired by Charikar (2000))

1. iteratively and greedily peel $V$ removing at each step $v = \arg\min_{v \in V} \text{score}(v, G[V])$
2. compute the density target density on the remaining network where $V = V \setminus \{v\}$
3. return the vertex-set $S \subseteq V$ maximizing the target density over all $O(n)$ iterations

How is $\text{score}(v, G[V])$ computed? Depends on the objective $f$

- for $f_{\text{min},\text{min}}(S, G)$ we set $\text{score}(v, G[V_i]) = \min_{t=1,\ldots,T} d(v, G_t[V])$
- for $f_{\text{avg},\text{avg}}(S, G)$ we set $\text{score}(v, G[V_i]) = \frac{1}{T} \sum_t d(v, G_t[V])$

Resulting algorithms run in $O(nT + \sum_t |E_t|)$

- FindBFF-MM ($f_{\text{min},\text{min}}$), finds the optimal solution
- FindBFF-AA ($f_{\text{avg},\text{avg}}$), finds a $\frac{1}{2}$-approximation
- BFF-AM ($f_{\text{avg},\text{min}}$) is $\text{NP}$-hard, FindBFF achieves an approximation at most $O(n^{-1})$
- BFF-MA ($f_{\text{min},\text{avg}}$), “complexity = ?”, FindBFF achieves an approximation at most $O(n^{-1/2})$
On the Complexity of BFF-MA

On this line of research Charikar et al. (2018) proved

- $f_{\min,\text{avg}}(S, G)$ cannot be approximated within $2^{\log^{1-\varepsilon} n}$ unless $\text{NP} \subseteq \text{DTIME}(n^{\text{poly log}(n)})$

- they give $\mathcal{O}((n \log T)^{-1/2})$ and $\mathcal{O}(n^{-2/3})$ approximation algorithms

- suppose $T$ is small the authors give and exact algorithm running in $\mathcal{O}(n^T \text{poly}(n, T))$, and a FPTAS that given an $\varepsilon > 0$ outputs a $(1 + \varepsilon)$-approximation in $\mathcal{O}(f(T)\text{poly}(n, \varepsilon^{-1}))$. 
FindBFF — A Use Case

Dataset consists of publications from DBLP in years from 2006 to 2015 (each year forms a snapshot \( G_i \)), the set of nodes \( V (|V| = 2625) \) represents authors.

Some observations

▶ Not all authors appearing in a dense subset coauthored many papers together, e.g., "Wei Fan", "Philip S. Yu", and "Jiawei Han" coauthored only two papers together but many pairwise.

▶ Some solutions are not connected, e.g., BFF-MM.
FindBFF — A Use Case

Dataset consists of publications from DBLP in years from 2006 to 2015 (each year forms a snapshot $G_i$), the set of nodes $V$ ($|V| = 2625$) represents authors.

<table>
<thead>
<tr>
<th>BFF-MM</th>
<th>BFF-MA</th>
<th>BFF-AM</th>
<th>BFF-AA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wei Fan, Philip S. Yu, Jiawei Han, Charu C. Aggarwal</td>
<td>Wei Fan, Jing Gao, Philip S. Yu, Jiawei Han, Charu C. Aggarwal</td>
<td>Wei Fan, Jing Gao, Philip S. Yu, Jiawei Han, Charu C. Aggarwal</td>
<td>Wei Fan, Jing Gao, Philip S. Yu, Jiawei Han, Charu C. Aggarwal, Mohammad M. Masud, Latifur Khan, Bhavani M. Thuraisingham</td>
</tr>
<tr>
<td>Lu Qin, Jeffrey Xu Yu, Xuemin Lin</td>
<td>Philip S. Yu, Jiawei Han, Charu C. Aggarwal</td>
<td>Philip S. Yu, Jiawei Han</td>
<td>Mohammad M. Masud, Latifur Khan, Bhavani M. Thuraisingham</td>
</tr>
<tr>
<td>Craig Macdonald, Iadh Ounis</td>
<td>Jeffrey Xu Yu, Xuemin Lin, Ying Zhang</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Some observations

- Not all authors appearing in a dense subset coauthored many papers together, e.g., “Wei Fan”, “Philip S. Yu”, and “Jiawei Han” coauthored only two papers together but many pairwise.

- Some solutions are not connected, e.g., BFF-MM.
Pros and Cons

Strengths
▶ computes dense communities based on degree scores
▶ it is efficient for certain formulations
▶ works well for sufficiently temporal course-grained data

What is missing
▶ not optimizing for specific time-frames of the time-span
▶ cannot adapt well for fine-grained data
▶ theoretical gaps may prevent (close to) optimal solutions

Lets see another formulation!
Communities as Episodes (Rozenshtein et al., 2019)

Data. Let $G = (V, E)$ be an undirected temporal graph with $E = \{(u_i, v_i, t_i): i = 1, \ldots, m\}$.

Setting. Find a multiple dense communities covering the timespan $[t_1, t_m]$. 

Communities as Episodes (Rozenshtein et al., 2019)

Data. Let $G = (V, E)$ be an undirected temporal graph with $E = \{(u_i, v_i, t_i) : i = 1, \ldots, m\}$.

Setting. Find a multiple dense communities covering the timespan $[t_1, t_m]$

Metrics and temporal properties.
Given an interval $I = [t_s, t_e] \subseteq [t_1, t_m]$ let $G[I] = (V[I], E[I])$ be the induced subgraph in $I$

An episode is a pair $(I = [t_s, t_e], H \subseteq G[I])$ where $H$ is a subgraph of $G[I]$

Given a static subgraph $H = (V_H, E_H)$ its density is $d(H) = \frac{|E_H|}{|V_H|}$
Communities as Episodes (Rozenshtein et al., 2019)

Data. Let $G = (V, E)$ be an undirected temporal graph with $E = \{(u_i, v_i, t_i) : i = 1, \ldots, m\}$.

Setting. Find a multiple dense communities covering the timespan $[t_1, t_m]$

Metrics and temporal properties.
Given an interval $I = [t_s, t_e] \subseteq [t_1, t_m]$ let $G[I] = (V[I], E[I])$ be the induced subgraph in $I$

An episode is a pair $(I = [t_s, t_e], H \subseteq G[I])$ where $H$ is a subgraph of $G[I]$

Given a static subgraph $H = (V_H, E_H)$ its density is $d(H) = \frac{|E_H|}{|V_H|}$

Problem

Given a temporal network $G$ and an integer $k$ find a set of $k$ episodes $(I_i, H_i), i = 1, \ldots, k$ such that $I_i \cap I_j = \emptyset, i \neq j$ are disjoint while maximizing

$$\sum_{i=1}^{k} d(H_i)$$
Solving the Problem through Segmentation

Computing the solution

- **Note!** in an optimal solution $\bigcup_i I_i$ covers $[t_1, t_m]$.
- given interval $I$ use result in (Goldberg, 1984) in $O(nm_{st} \log n)$ to obtain $H^* = \arg \max_{H \subseteq G[I]} d(H)$, and $m_{st}$ is the max edges in an interval
- it suffices to find an optimal segmentation $I_1, \ldots, I_k$, (with dyn. prog. in $O(kT^2)$)

Such algorithm has runtime $O(kT^2 nm_{st} \log n)$
Solving the Problem through Segmentation

Computing the solution

- **Note!** in an optimal solution $\bigcup_i I_i$ covers $[t_1, t_m]$.
- given interval $I$ use result in (Goldberg, 1984) in $O(n m_{st} \log n)$ to obtain $H^* = \arg \max_{H \subseteq G[I]} d(H)$, and $m_{st}$ is the max edges in an interval
- it suffices to find an **optimal segmentation** $I_1, \ldots, I_k$, (with dyn. prog. in $O(k T^2)$)

Such algorithm has runtime $O(k T^2 n m_{st} \log n)$

As this can be prohibitive an **approximate approach** is proposed

- **approximate dyn. prog.** approach controlled $\varepsilon_1$ pruning candidates in segmentation
- avoiding recomputing densest subgraph, use **evolving solution**, controlled by $\varepsilon_2$

Leading to $2(1 + \varepsilon_1)(1 + \varepsilon_2)$-approximation algorithm running in $O(k^2 \frac{T m_{T,\text{max}} \log^2(n)}{\varepsilon_1 \varepsilon_2})$ where $m_{T,\text{max}}$ is the maximum number of edges with the same time-stamp.
Communities as Episodes — Analyzing Twitter Data

Tweets in Helsinki region, $V$ are hashtags and $(u, v, t)$ is added if a message contains both hashtags $u, v$, data gathered from Nov 2013 containing 4758 tweets and 917 nodes. (Parameters, $k = 4$ and $\varepsilon_1 = \varepsilon_2 = 0.1$)
Communities as Episodes — Analyzing Twitter Data

Tweets in Helsinki region, $V$ are hashtags and $(u, v, t)$ is added if a message contains both hashtags $u, v$, data gathered from Nov 2013 containing 4758 tweets and 917 nodes. (Parameters, $k = 4$ and $\varepsilon_1 = \varepsilon_2 = 0.1$)

Both episodes are related to events occurring in that period in Helsinki and globally, e.g., on the left the Digiexpo and Halloween events, while on the right the MTV Europe music awards that was on November 10.
Tweets in Helsinki region, $V$ are hashtags and $(u, v, t)$ is added if a message contains both hashtags $u, v$, data gathered from Nov 2013 containing 4758 tweets and 917 nodes. (Parameters, $k = 4$ and $\varepsilon_1 = \varepsilon_2 = 0.1$)

- both episodes are related to events occurring in that period in Helsinki and globally, e.g., on the left the Digiexpo and Halloween events, while on the right the MTV Europe music awards that was on November 10.
Pros and Cons

Strengths
▶ it adapts in a flexible way to the timespan of the network
▶ in general it is efficient to compute
▶ identifies dense communities (w.r.t., average degree)

What is missing
▶ when aggregation is performed may lose some information
▶ not accounting for temporal behaviors, e.g., periodicity or burstiness
▶ may not be informative with small timespan (i.e, $G$ has not few dense structures)

Lets see another formulation!
What about Social Networks
Discovering Buzzing Stories (Bonchi et al., 2019)

**Data.** Given a temporal network \( G = (V, \{E_t, f_t\}_{t=1,\ldots,T}) \).
Discovering Buzzing Stories (Bonchi et al., 2019)

Data. Given a temporal network $G = (V, \{E_t, f_t\}_{t=1,...,T})$.

- $V$ is a set of objects, e.g., *messages or words* from chat apps or posts on socials.

- $E_t$ edges at time $t$ and $f_t : E_t \rightarrow \mathbb{R}^+$ is a weighting function capturing the strength between two objects at $t$, e.g., $f_t(\text{cream, carbonara})$
Discovering Buzzing Stories (Bonchi et al., 2019)

Data. Given a temporal network $G = (V, \{E_t, f_t\}_{t=1,\ldots,T})$.

Setting. Find multiple dense and unexpected stories, i.e., communities.

Metrics and temporal properties.

Transform $G \rightarrow G_A = (V, \{E_t, \phi_t\}_{t=1,\ldots,T})$ by mapping $f_t \rightarrow \phi_t$ where $\phi_t(\cdot)$ captures how anomalous or unexpected is $e$ based on past data.

Given a discrete-time interval $I = [t_s, t_e]$ and $S \subseteq G_A$ then “$\delta(S, I) =$ density of $S$ in $I$”

Given a set of subgraphs $S$ then $\Delta(S, I) = \sum_{S \in S} \delta(S, I)$.
Discovering Buzzing Stories (Bonchi et al., 2019)

Data. Given a temporal network $G = (V, \{E_t, f_t\}_{t=1,\ldots,T})$.

Setting. Find multiple dense and unexpected stories, i.e., communities.

Metrics and temporal properties.

Transform $G \rightarrow G_A = (V, \{E_t, \phi_t\}_{t=1,\ldots,T})$ by mapping $f_t \rightarrow \phi_t$ where $\phi_t(\cdot)$ captures how anomalous or unexpected is $e$ based on past data.

Given a discrete-time interval $I = [t_s, t_e]$ and $S \subseteq G_A$ then “$\delta(S, I) = \text{density of } S \text{ in } I$”

Given a set of subgraphs $S$ then $\Delta(S, I) = \sum_{S \in S} \delta(S, I)$.

Problem

Given the temporal (anomalous) graph $G_A$, an interval $I \subseteq [t_1, t_m]$ and two integers $K, N \geq 1$ find $S^* = \{S_1, \ldots, S_K\}$ of disjoint subgraphs of $G_A$ such that

- $|S_i| \leq N$, that is the number of nodes in each subgraph is bounded by $N$
- $\Delta(S^*, I)$ is maximized.
Discovering Buzzing Stories — Algorithm

The decision problem is \textbf{NP}-hard.

Computing a solution

- develop an exact algorithm $A$ running in $O(|I|m \log n)$ for $K = 1$ and $N = \infty$ based on a “peeling” the minimum degree vertices.

\textbf{Note!} This needs to be accounting for time-points (updating the vertex degrees).
Discovering Buzzing Stories — Algorithm

The decision problem is \textbf{NP}-hard.

Computing a solution

- develop an exact algorithm $A$ running in $O(|I|m \log n)$ for $K = 1$ and $N = \infty$ based on a “peeling” the minimum degree vertices.

\textbf{Note!} This needs to be accounting for time-points (updating the vertex degrees).

- algorithm for general case using $A$ as subroutine by
  - imposing further \textit{constraints on the size} (i.e., bounding $N$) of the reported solutions
  - iteratively \textit{remove identified communities} to guarantee disjoint output.

The resulting algorithms runs in $O(K|I|m \log n)$ and comes without guarantees.
Discovering Buzzing Stories – A Use Case

**Dataset.** Yahoo searches during 2013-2014

**Assumption.** If there is an anomaly people will search it on the web!

**Processing.**

- dataset spans 558 days, is build on user queries (appearing at least 50 times per day)
- map queries e.g., "How to put pineapple on pizza" → "(pineapple,pizza)" is generated
- \( f_t \) accounts for frequency

<table>
<thead>
<tr>
<th>Date</th>
<th>I</th>
<th>N</th>
<th>Story Related Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>13/01/2014</td>
<td>1</td>
<td>10</td>
<td>&quot;cristiano dor wins ronaldo fifa ballon&quot; Cristiano Ronaldo won the Ballon d'Or in 2014</td>
</tr>
<tr>
<td>09/02/2014</td>
<td>3</td>
<td>10</td>
<td>&quot;day figure russia julia skating medal ceremony&quot; Yulia Lipnitskaya, a Russian prodigy won gold medal in Sochi</td>
</tr>
<tr>
<td>27/02/2014</td>
<td>2</td>
<td>30</td>
<td>&quot;captains costa wreck concordia&quot; Legal process for the Costa Concordia disaster</td>
</tr>
</tbody>
</table>
Discovering Buzzing Stories – A Use Case

**Dataset.** Yahoo searches during 2013-2014

**Assumption.** If there is an anomaly people will search it on the web!

**Processing.**
- dataset spans 558 days, is built on user queries (appearing at least 50 times per day)
- map queries e.g., ”How to put pineapple on pizza” $\rightarrow”(\text{pineapple},\text{pizza})”$ is generated
- $f_t$ accounts for frequency

| Date       | $|I|$ | $N$ | Story                                      | Related Event                                           |
|------------|-----|-----|--------------------------------------------|--------------------------------------------------------|
| 13/01/2014 | 1   | 10  | “cristiano dor wins ronaldo fifa ballon”   | Cristiano Ronaldo won the Ballon d’Or in 2014          |
| 09/02/2014 | 3   | 10  | “day figure russia julia skating medal ceremony” | Yulia Lipnitskaya, a Russian prodigy won gold medal in Sochi |
| 27/02/2014 | 2   | 30  | “captains costa wreck concordia”           | Legal process for the Costa Concordia disaster          |
Pros and Cons

Strengths
- tailored to a specific application
- interesting analyses
- efficient algorithm in practice

What is missing
- no guarantees
- a lot of preprocessing is needed and identifying $\phi(\cdot)$ may be non-trivial
- not much used in practice
TIME FOR ...

QUESTIONS
Significant Engagement Based Community (Zhang et al., 2022)

Dataset. Undirected temporal network $G = (V, E), E = \{(u_i, v_i, t_i) : i = 1, \ldots, m\}$

Setting. Find the community where a given user has highest engagement, local formulation!
Significant Engagement Based Community (Zhang et al., 2022)

Dataset. Undirected temporal network $G = (V, E), E = \{(u_i, v_i, t_i) : i = 1, \ldots, m\}$

Setting. Find the community where a given user has highest engagement, local formulation.

Metrics and temporal properties. Let $H \subseteq G$, the degree of $v \in V[H]$ is $d_{u, H} = \sum_{e \in E[H]} 1[v \in e]$. Given $H \subseteq G$ the engagement of $v \in V[H]$ is $\gamma(u, H) = \frac{d_{u, H}}{\sum_{v \in V[H]} d_{v, H}}$. 
Significant Engagement Based Community (Zhang et al., 2022)

Dataset. Undirected temporal network $G = (V, E), E = \{(u_i, v_i, t_i) : i = 1, \ldots, m\}$

Setting. Find the community where a given user has highest engagement, local formulation!

Metrics and temporal properties.
Let $H \subseteq G$, the degree of $v \in V[H]$ is $d_{u,H} = \sum_{e:E[H]} 1[v \in e]$.

Given $H \subseteq G$ the engagement of $v \in V[H]$ is $\gamma(u, H) = \frac{d_{u,H}}{\sum_{v \in V[H]} d_{v,H}}$.

Problem

Given a temporal graph $G$, a parameter $k \geq 1$, and a vertex $u \in V$ find $H$ such that
- $u \in V[H]$
- the static network of $H$ is a $k$-core (Guarantees cohesiveness)
- $\gamma(u, H) \geq \gamma(u, H')$ for any other $H' \subseteq G$ (Guarantees max-engagement)
A Case Study on DBLP (Zhang et al., 2022)

Computing a solution

Greedy peeling algorithms + local search running in $O(m^2(n + m))$
A Case Study on DBLP (Zhang et al., 2022)

Computing a solution

Greedy peeling algorithms + local search running in $O(m^2(n + m))$

Insights on DBLP data

As desired seed nodes are well centered in the identified communities
Pros and Cons

Strengths

▶ local formulation
▶ proposed algorithm is polynomial
▶ output has desired properties

What is missing

▶ no guarantees
▶ may need additional assumptions to better model engagement
▶ engagement is not time-dependent
You showed us only degree-based metrics!
Local Motif Clustering (Fu et al., 2020)

Data. Undirected temporal network $G = \{G_1, \ldots, G_T\}$.

Setting. Find a good local “tight cluster $\rightarrow$ motifs” to a seed node at each time point $1, \ldots, T$. 

Data.  Undirected temporal network $G = \{G_1, \ldots, G_T\}$.  

Setting.  Find a good local “tight cluster $\rightarrow$ motifs” to a seed node at each time point $1, \ldots, T$.  

Metrics and temporal properties. $H$: a small subgraph pattern (e.g., edge, triangle, star etc). $C \subseteq V$: is cluster, and motif-conductance is $\Phi(C, H) = \min\{\text{vol}(C, H), \text{vol}(\overline{C}, H)\}$ where,  

- $\partial(C, H)$: # of subgraphs cut by the cluster (i.e., “cut = at least one node not in $C$”)  
- $\text{vol}(A, H)$: # of occurrences of $H$ in $A$
Local Motif Clustering (Fu et al., 2020)

Data. Undirected temporal network \( G = \{ G_1, \ldots, G_T \} \).

Setting. Find a good local “tight cluster \( \rightarrow \) motifs” to a seed node at each time point \( 1, \ldots, T \).

Metrics and temporal properties.
\( H \): a small subgraph pattern (e.g., edge, triangle, star etc).
\( C \subseteq V \): is cluster, and motif-conductance is
\[
\Phi(C, H) = \frac{\partial(C, H)}{\min\{\text{vol}(C, H), \text{vol}(\overline{C}, H)\}}
\]
where,
- \( \partial(C, H) \): \# of subgraphs cut by the cluster (i.e., “cut = at least one node not in \( C \)”)
- \( \text{vol}(A, H) \): \# of occurrences of \( H \) in \( A \)
Local Motif Clustering (Fu et al., 2020)

Data. Undirected temporal network $G = \{G_1, \ldots, G_T\}$.

Setting. Find a good local “tight cluster $\rightarrow$ motifs” to a seed node at each time point $1, \ldots, T$.

Metrics and temporal properties.
$H$: a small subgraph pattern (e.g., edge, triangle, star etc).
$C \subseteq V$: is cluster, and motif-conductance is $\Phi(C, H) = \frac{\partial(C, H)}{\min\{vol(C, H), vol(\bar{C}, H)\}}$ where,

- $\partial(C, H)$: # of subgraphs cut by the cluster (i.e., “cut = at least one node not in $C$”)
- $vol(A, H)$: # of occurrences of $H$ in $A$

Problem

Given $G = \{G_1, \ldots, G_T\}$, a static motif $H$, a seed node $v \in V$, an upper-bound on the motif conductance $\phi$, compute,

$C^t$ containing $v \in V$ such that $\Phi(C^t, H) \leq \phi$, for each $t = 1, \ldots, T$. 

Local Motif Clustering – Example

Example

Let us fix $H$ to be a triangle

**Notation.** Blue edges denote insertions and green edges denote removals.

(a) $t = 1$
Local Motif Clustering – Example

Example

Let us fix $H$ to be a triangle

**Notation.** Blue edges denote insertions and green edges denote removals.
Let us fix $H$ to be a triangle.

**Notation.** Blue edges denote insertions and green edges denote removals.

(a) $t = 1$

(b) $t = 2$

(c) $t = 3$
Proposed Solution by Fu et al. (2020)

The problem is already for static graphs NP-hard 😞
Proposed Solution by Fu et al. (2020)

The problem is already for static graphs \textbf{NP}-hard 😞

We will be looking for an \textit{approximate solution}
Proposed Solution by Fu et al. (2020)

The problem is already for static graphs \textbf{NP-hard} 😞

Computing a solution

- fix $t = 1, \ldots, T$, let $k$ be \# nodes in $H$
- define a multilinear page-rank vector $x_t$, that accounts for the high-order motifs,

$$x_t = \alpha P_t(x_t \otimes \cdots \otimes x_t) + (1 - \alpha)u$$

where $P_t$ encodes transitions over motifs, $u$ is the vector encoding user preferences.
Proposed Solution by Fu et al. (2020)

The problem is already for static graphs \( \textbf{NP}-\text{hard} \)

Computing a solution

- fix \( t = 1, \ldots, T \), let \( k \) be \# nodes in \( H \)
- define a multilinear page-rank vector \( x^t \), that accounts for the high-order motifs,

\[
x^t = \alpha P^t(x^t \otimes \cdots \otimes x^t) + (1 - \alpha)u
\]

where \( P^t \) encodes transitions over motifs, \( u \) is the vector encoding user preferences.
- a good-approximate obtained through \textit{sweep cut} on vector \( x^t \),
  - sort (from largest to smallest) the entries in \( x^t \)
  - pick the prefix \( 1, \ldots, j \) with \( j = 1, \ldots, n \) optimizing the induced cut
Proposed Solution by Fu et al. (2020)

The problem is already for static graphs \( \text{NP-hard} \)

Computing a solution

- fix \( t = 1, \ldots, T \), let \( k \) be \# nodes in \( H \)
- define a multilinear page-rank vector \( x^t \), that accounts for the high-order motifs,

\[
x^t = \alpha P^t(x^t \otimes \cdots \otimes x^t) + (1 - \alpha) u
\]

where \( P^t \) encodes transitions over motifs, \( u \) is the vector encoding user preferences.

- a good-approximate obtained through sweep cut on vector \( x^t \),
  - sort (from largest to smallest) the entries in \( x^t \)
  - pick the prefix \( 1, \ldots, j \) with \( j = 1, \ldots, n \) optimizing the induced cut

**Idea!** For varying \( t \), graph is evolving avoid repeating such steps by 1. avoid considering distant edges that cannot impact cluster 2. avoid scratch re computation of \( x^t \)

Runtime \( \mathcal{O}(\sum_t[f(m_i, n_i^{O(k)}, k^k)]) \), \( P^1 \) is assumed in input!
Pros and Cons

Strengths

▶ local formulation
▶ uses high order information
▶ versatile according to the pattern

What is missing

▶ no guarantees
▶ problem is already hard on a single snapshot
▶ not very practical computing \( P^i, i = 1, \ldots \)
Fairness-Aware Clique Preserving Clustering (Fu et al., 2023)

Fairness definition demographic fairness

\[ G = \{ G_1, \ldots, G_T \} \]

The problem is \( \text{NP}-\text{hard} \).

At fixed \( t \), the solution is based on spectral techniques: trace minimization problem + \( K \)-means.

Avoid recomputation at each time \( t \), accounting for edge additions and deletions.

Runtime is \( O(T(q^{4} + q^{2}n)) + P_{t}ka^{k-2}m_t) \), where \( a \) is the arboricity at time \( t \), with no guarantees on the solution.
Fairness-Aware Clique Preserving Clustering (Fu et al., 2023)

Fairness definition demographic fairness
Data. $G = \{G_1, \ldots, G_T\}$, where $V$ has $h$ different groups

Setting. Find $q$ clusters at each time-point cutting few $k$-cliques and that are fair.

**Problem**

Given $G$ parameters $k$ and $q$, find a $q$-clustering $C^t_1, \ldots, C^t_q$ for $t = 1, \ldots, T$ such that

- $\min_{C^t_i} \sum_{t=1}^T \sum_{j=1}^q \frac{\partial(C^t_i, k)}{\text{vol}(C^t_i, k)}$ ($\partial(C^t_i, k)$: number of $k$-cliques cut by $C^t_i$)

- $\frac{|V_s \cap C^t_i|}{|C^t_i|} = \frac{|V_s|}{|V|}$ for each time $t = 1, \ldots, T$ and cluster $s = 1, \ldots, q$
Fairness-Aware Clique Preserving Clustering (Fu et al., 2023)

Data. \( G = \{ G_1, \ldots, G_T \} \), where \( V \) has \( h \) different groups

Setting. Find \( q \) clusters at each time-point cutting few \( k \)-cliques and that are fair.

Problem

Given \( G \) parameters \( k \) and \( q \), find a \( q \)-clustering \( C^t_1, \ldots, C^t_q \) for \( t = 1, \ldots, T \) such that

\[
\begin{align*}
\min_{C^t_i} & \sum_{t=1}^T \sum_{j=1}^q \frac{\partial(C^t_i, k)}{\text{vol}(C^t_i, k)} \left( \partial(C^t_i, k): \text{# of } k\text{-cliques cut by } C^t_i \right) \\
\text{and} & \quad \frac{|V_s \cap C^t_i|}{|C^t_i|} = \frac{|V_s|}{|V|} \text{ for each time } t = 1, \ldots, T \text{ and cluster } s = 1, \ldots, q
\end{align*}
\]

Computing a solution

The problem is \textbf{NP}-hard

At fixed \( t \) solution is based on \textit{spectral techniques}: trace minimization problem + \( K \)-means.

Avoid re computation at each time \( t \), accounting for edge additions and deletions.

Runtime is \( O(T(q^4 + q^2 n) + \sum_t ka^{k-2} m^t) \), \( a \) is arboricity at time \( t \), no guarantees on the solution.
Visualizing the Desired Behavior

Example
Let $q = 2$ and $k = 3$ ($k$-clique is a triangle). **Green edges** are insertions and **yellow edges** are removals.
**Visualizing the Desired Behavior**

**Example**

Let $q = 2$ and $k = 3$ ($k$-clique is a triangle). *Green edges* are insertions and *yellow edges* are removals.
Pros and Cons

Strengths
▶ fairness + evolving networks
▶ global dense clusters
▶ can have different applications

What is missing
▶ no guarantees
▶ problem is already hard on a single snapshot
▶ not very practical enumerating $k$-cliques
OK, OK, ENOUGH ALREADY!
OK, OK, ENOUGH ALREADY!

Let us do a summary.
► **community detection in temporal networks** is a very wide research area
► do not panic and follow a **principled approach** (start from data!)
  – identify **properties of the communities** you are looking (global vs. local, etc..)
  – **search formulations** with desired properties (*much work has been done!*)
  – if nothing works you found a **gap in literature** (novel algorithms are needed!)
► **use/develop proper algorithms** to analyze temporal communities

**Keep in mind.** There is a **gap between formulations and applications**
Temporal communities – Other Formulations

Some other existing formulations

- (Lin et al., 2022) find multiple maximal quasi-clique based communities, stable overall and with interval-based edges
- (Qin et al., 2023) find single and dense community that is periodic over time
- (Preti et al., 2021) discovering a set of diverse and correlated communities in dynamic setting
- (Ma et al., 2020) finding dense subgraphs in temporal networks with time-varying edge weight
- (Banerjee and Pal, 2022) online algorithm for temporal clique identification
- (Chu et al., 2019) bursty and dense community identification
- . . .
Coffee time

Right meow
Temporal Motifs and Events
Motifs are small subgraph patterns with a plethora of applications in various domains.

- databases
- social networks
- biology
- e-commerce
Motifs are small subgraph patterns with a plethora of applications in various domains.

- databases
  - Cassandra
  - SAP
  - HBase
  - IBM DB2
  - ORACLE
  - SQL Server
  - SQLite
  - Amazon
  - Oracle
  - MySQL
  - InfluxDB
  - MariaDB

- social networks
  - Facebook
  - Instagram
  - Twitter
  - TikTok
  - YouTube
  - LinkedIn

- biology
  - Illumina
  - Roche
  - Life Technologies
  - Ion Torrent
  - Pacific Biosciences
  - Helicos
  - Nanopore

- e-commerce
  - Amazon
  - Macy's
  - eBay
  - Target
  - Best Buy
  - Alibaba
  - Etsy
  - H&M
  - Walmart
  - Chegg
  - Wayfair

But how?
Short Primer on Subgraph Isomorphism

Given a simple static graph \( G = (V, E) \) and small target graph \( H = (V_H, E_H) \) we say that,

\[ G' \subseteq G \text{ is a subgraph that is isomorphic to } H \text{ if there exists a bijection } f : V_H \leftrightarrow V_{G'} \text{ with} \]

\[ (x, y) \in E_H \implies (f(x), f(y)) \in E_{G'} \quad (G' \sim H) \]
Short Primer on Subgraph Isomorphism

Given a simple static graph $G = (V, E)$ and small target graph $H = (V_H, E_H)$ we say that,

$G' \subseteq G$ is a subgraph that is isomorphic to $H$ if there exists a bijection $f : V_H \mapsto V_{G'}$ with

$$(x, y) \in E_H \implies (f(x), f(y)) \in E_{G'} \quad (G' \sim H)$$

Example

Is “$V[C, D, E]$” isomorphic to $H$?

![Diagram](image-url)
Short Primer on Subgraph Isomorphism

Given a simple static graph $G = (V, E)$ and small target graph $H = (V_H, E_H)$ we say that,

$G' \subseteq G$ is a subgraph that is isomorphic to $H$ if there exists a bijection $f : V_H \mapsto V_{G'}$ with

$$(x, y) \in E_H \implies (f(x), f(y)) \in E_{G'}, \quad (G' \sim H)$$

Example

Is “$V[C, D, E]$” isomorphic to $H$? Yes!
Short Primer on Subgraph Isomorphism

Given a simple static graph $G = (V, E)$ and small target graph $H = (V_H, E_H)$ we say that,

$\begin{align*}
G' &\subseteq G \text{ is a subgraph that is isomorphic to } H \text{ if there exists a bijection } f : V_H \leftrightarrow V_{G'} \text{ with } \\
(x, y) &\in E_H \implies (f(x), f(y)) \in E_{G'} \quad (G' \sim H)
\end{align*}$

Example

Is “$V[A, C, B]$” isomorphic to $H$?

(a) $G$

(b) $H$

392
Short Primer on Subgraph Isomorphism

Given a simple static graph $G = (V, E)$ and small target graph $H = (V_H, E_H)$ we say that,

$\triangleright\ G' \subseteq G$ is a subgraph that is isomorphic to $H$ if there exists a bijection $f : V_H \leftrightarrow V_{G'}$ with

$$(x, y) \in E_H \implies (f(x), f(y)) \in E_{G'}, \quad (G' \sim H)$$

Example


(a) $G$

(b) $H$
Short Primer on Subgraph Isomorphism

Given a simple static graph \( G = (V, E) \) and small target graph \( H = (V_H, E_H) \) we say that,

\( G' \subseteq G \) is a subgraph that is isomorphic to \( H \) if there exists a bijection \( f : V_H \mapsto V_{G'} \) with

\[
(x, y) \in E_H \iff (f(x), f(y)) \in E_{G'} \quad (G' \sim H)
\]

the isomorphism is called *induced* if it also holds

\[
(x, y) \in E_H \iff (f(x), f(y)) \in E_{G'}
\]

Example

Is “\( V[A, C, B] \)” isomorphic to \( H \)?

\[\begin{array}{ll}
(\text{a) } G & (\text{b) } H \\
(\text{c) } H & (\text{d) } \text{Isomorphism to } H?}
\end{array}\]
Short Primer on Subgraph Isomorphism

Given a simple static graph $G = (V, E)$ and small target graph $H = (V_H, E_H)$ we say that,

- $G' \subseteq G$ is a subgraph that is isomorphic to $H$ if there exists a bijection $f : V_H \mapsto V_{G'}$ with

  $$(x, y) \in E_H \implies (f(x), f(y)) \in E_{G'} \quad (G' \sim H)$$

- the isomorphism is called *induced* if it also holds

  $$(x, y) \in E_H \iff (f(x), f(y)) \in E_{G'}$$

Example

Is "$V[A, C, B]$" isomorphic to $H$? **No!**

\[ (f^{-1}(A), f^{-1}(B)) \text{ not in } H! \]
Short primer on Subgraph Isomorphism

Given a simple static graph \( G = (V, E) \) and small target graph \( H = (V_H, E_H) \) we say that,

\[ G' \subseteq G \text{ is a subgraph that is isomorphic to } H \text{ if there exists a bijection } f : V_H \rightarrow V_{G'} \text{ with} \]

\[ (x, y) \in E_H \implies (f(x), f(y)) \in E_{G'} \quad (G' \sim H) \]

\[ \text{the isomorphism is called } \textit{induced} \text{ if it also holds} \]

\[ (x, y) \in E_H \iff (f(x), f(y)) \in E_{G'} \]

Given a graph \( G \), \textit{count} of \( H \) meaning: \# of distinct subgraphs \( G' \subseteq G \) with \( G' \sim H \).

If \( G' \sim H \) we say \( G' \) is an \textit{occurrence} of \( H \)

**Problem**

Given a graph \( G \) and a \textit{small} subgraph pattern \( H \)

\[ \text{obtain the count of } H \text{ (counting problem)} \]

\[ \text{list all occurrences of } H \]
Subgraph Counts

The problem is **NP-hard** and extremely challenging.

Many applications, in *computer science, network science* and more...

(a) Node embeddings
Subgraph Counts

The problem is \textbf{NP}-hard and extremely challenging.

Many applications, in \textit{computer science}, \textit{network science} and more...

(a) Node embeddings

(b) Malware detection
Subgraph Counts

The problem is \textbf{NP}-hard and extremely challenging

Many applications, in \textit{computer science, network science} and more…

(a) Node embeddings  
(b) Malware detection  
(c) Spreading processes

Some material

- (Ribeiro et al., 2021) survey on algorithms and applications
- (Seshadhri and Tirthapura, 2019) tutorial in WWW 2019 on subgraph counting

\textbf{What about temporal motifs?}
Temporal Motifs

As for temporal communities many definitions exist:

- Temporal motifs = static subgraphs + temporal dynamics (+ additional information)
  - ▶ static subgraph may be (non)induced
  - ▶ temporal dynamics time over the static subgraph, in many ways
  - ▶ additional information is any available metadata, e.g., $f: \{V, E\} \rightarrow D$

Let us see some of the most used definitions.
Temporal Motifs

As for temporal communities many definitions exist

\[
\text{Temporal motifs} = \text{static subgraphs} + \text{temporal dynamics} (+ \text{additional information})
\]

Where

- static subgraph may be (non)induced

- temporal dynamics time over the static subgraph, in many ways
Temporal Motifs

As for temporal communities many definitions exist

Temporal motifs = static subgraphs + temporal dynamics (+ additional information)

Where

▸ static subgraph may be (non)induced

▸ temporal dynamics time over the static subgraph, in many ways

▸ additional information is any available metadata on nodes or edges, e.g., $f : \{V, E\} \rightarrow D$

Let us see some of most used definitions
Temporal Motifs by Kovanen et al. (2011)

**Data.** $G = (V, E), E = \{(u, v, t)\}$ be a *directed* temporal network

Modeling temporal dynamics

Some definitions

- two edges $(e_1, e_2 \in E)$ are $\Delta t$-adjacent they share at least one node and $|t_{e_1} - t_{e_2}| \leq \Delta t$. 
Temporal Motifs by Kovanen et al. (2011)

**Data.** $G = (V, E), E = \{(u, v, t)\}$ be a directed temporal network

**Modeling temporal dynamics**

**Some definitions**

- Two edges $(e_1, e_2 \in E)$ are $\Delta t$-adjacent if they share at least one node and $|t_{e_1} - t_{e_2}| \leq \Delta t$.

**Example.** Fix $\Delta t = 10$

(a) Temporal Network $G$

(b) $\Delta t$-adjacent edges

(c) Not $\Delta t$-adjacent
Data. $G = (V,E), E = \{(u,v,t)\}$ be a directed temporal network

Modeling temporal dynamics

Some definitions

- two edges $(e_1, e_2 \in E)$ are $\Delta t$-adjacent they share at least one node and $|t_{e_1} - t_{e_2}| \leq \Delta t$.
- two edges $(e_1, e_2 \in E)$ are $\Delta t$-connected if there is a sequence $\Delta t$-adj. edges from $e_1$ to $e_2$ (no time ordering!)
Temporal Motifs by Kovanen et al. (2011)

**Data.** $G = (V, E), E = \{(u, v, t)\}$ be a *directed* temporal network

**Modeling temporal dynamics**

Some definitions

- two edges $(e_1, e_2 \in E)$ are $\Delta t$-adjacent they share at least one node and $|t_{e_1} - t_{e_2}| \leq \Delta t$.
- two edges $(e_1, e_2 \in E)$ are $\Delta t$-connected if there is a sequence $\Delta t$-adj. edges from $e_1$ to $e_2$ (no time ordering!)

**Example.** Fix $\Delta t = 10$

![Temporal Network](image1)

(a) Temporal Network $G$

(b) $\Delta t$-connected edges
Data. $G = (V, E)$, $E = \{(u, v, t)\}$ be a directed temporal network

Modeling temporal dynamics

Some definitions

- two edges $(e_1, e_2 \in E)$ are $\Delta t$-adjacent if they share at least one node and $|t_{e_1} - t_{e_2}| \leq \Delta t$.
- two edges $(e_1, e_2 \in E)$ are $\Delta t$-connected if there is a sequence $\Delta t$-adj. edges from $e_1$ to $e_2$ (no time ordering!)

Further, a connected temporal subgraph $G' \subseteq G$ constitutes of pairwise $\Delta t$-conn. edges

- a conn. temporal subgraph $G'$ is valid if for each two events that are incident to a node no other edges are skipped (temporally induced $G'$)
Temporal Motifs by Kovanen et al. (2011)

**Data.** $G = (V, E), E = \{(u, v, t)\}$ be a *directed* temporal network

**Modeling temporal dynamics**

**Some definitions**
- two edges $(e_1, e_2 \in E)$ are $\Delta t$-*adjacent* they share at least one node and $|t_{e_1} - t_{e_2}| \leq \Delta t$.
- two edges $(e_1, e_2 \in E)$ are $\Delta t$-*connected* if there is a sequence $\Delta t$-adj. edges from $e_1$ to $e_2$ (no time ordering!)

Further, a *connected temporal subgraph* $G' \subseteq G$ constitutes of pairwise $\Delta t$-conn. edges
- a conn. temporal subgraph $G'$ is *valid* if for each two events that are incident to a node no other edges are skipped (temporally induced $G'$)

**Example.** Fix $\Delta t = 10$

![Temporal Network $G$](image-a)

![Valid](image-b)

![Not valid](image-c)
Temporal Motifs by (Kovanen et al., 2011)

**Definition**

Temporal motifs are *non-isomorphic* classes of subgraphs, where the isomorphism *takes into account edge ordering*.

**Example.**

(a) Motif occurrence

(b) Class 1?

(c) Class 2?
Temporal Motifs by (Kovanen et al., 2011)

**Definition**

Temporal motifs are non-isomorphic classes of subgraphs, where the isomorphism takes into account edge ordering.

**Example.**

(a) Motif occurrence

(b) Class 1? ✓

(c) Class 2? ×
Temporal Motifs by (Kovanen et al., 2011)

**Definition**

Temporal motifs are *non-isomorphic* classes of subgraphs, where the isomorphism *takes into account edge ordering*.

**Problem**

Given $G$, $\Delta t$ and a bound $k$ obtain the count of temporal motifs on $k$ nodes.

Solving the counting problem for specific classes

- pre-process $G$ and identify maximal components $O(|E|)$
- in each component find valid subgraphs of bounded size $k$, $O(n^k)$
- map each valid subgraph $G'$ with $k$-edges on its class (*canonical labeling* is used, exponential in $|G'|$).

A real use case?
Homophily in Phone Call Networks (Kovanen et al., 2013)

Data. Record of 6 months data of mobile phone calls (625 million calls) and SMS (207 million). In total > 6 million users.

Additional metadata. Sex, age and payment type, combined to obtain 24 different node colors.
Homophily in Phone Call Networks (Kovanen et al., 2013)

Data. Record of 6 months data of mobile phone calls (625 million calls) and SMS (207 million). In total > 6 million users.

Additional metadata. Sex, age and payment type, combined to obtain 24 different node colors.

Insights. Temporal homophily occurs at gender level, i.e., temporal phone call patterns tend to be different in males and females.

Temporal motifs.

The score denotes how strong is data with respect to a random-model. $F$: female, $M$: male, $F - F$: all nodes are of $F$ class, $F - *$: there exist at least one node of $M$ class in the motif.
Homophily in Phone Call Networks (Kovanen et al., 2013)

**Data.** Record of 6 months data of mobile phone calls (625 million calls) and SMS (207 million). In total > 6 million users.

**Additional metadata.** Sex, age and payment type, combined to obtain 24 different node colors

**Insights.** Temporal homophily occurs at gender level, i.e., temporal phone call patterns tend to be different in males and females

**Temporal motifs.**

![Temporal motifs diagram](image)

The score denotes how strong is data with respect to a random-model. $F$: female, $M$: male, $F - F$: all nodes are of $F$ class, $F - *$: there exist at least one node of $M$ class in the motif.

<table>
<thead>
<tr>
<th>Motif</th>
<th>$F - F$</th>
<th>$F - *$</th>
<th>$M - M$</th>
<th>$M - *$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeated calls</td>
<td>1.11</td>
<td>1.11</td>
<td>1.13</td>
<td>1.10</td>
</tr>
<tr>
<td>Noncausal chain</td>
<td><strong>1.08</strong></td>
<td><strong>1.02</strong></td>
<td>1.01</td>
<td>1.04</td>
</tr>
<tr>
<td>Causal chain</td>
<td><strong>1.08</strong></td>
<td><strong>1.01</strong></td>
<td><strong>0.98</strong></td>
<td><strong>1.02</strong></td>
</tr>
<tr>
<td>Out-star</td>
<td><strong>1.10</strong></td>
<td><strong>1.03</strong></td>
<td><strong>1.01</strong></td>
<td><strong>1.04</strong></td>
</tr>
</tbody>
</table>
An Issue with Kovanen et al. (2011)’s Definition

**Example.** Fix $\Delta t = 10$

(a) Temporal Network $G$

(b) Not valid
An Issue with Kovanen et al. (2011)’s Definition

Example. Fix $\Delta t = 10$

Consider only valid subgraphs may be too strict!
Which can lead to information loss, as this can be important for many applications.

How to fix this?
Temporal Motifs by Paranjape et al. (2017)

This model aims at providing a more general and flexible definition of temporal motifs.

**Data.** $G = (V, E), E = \{(u, v, t)\}$ be a directed temporal network

A **temporal motif** is a pair $M = (K, \sigma)$ (Liu et al., 2019) where

- $K$ is a directed and (weakly)connected multigraph with $k$-nodes and $\ell$-edges.

\[ v_1 \rightarrow v_2 \rightarrow v_3 \]
\[ v_1 \rightarrow v_2 \leftarrow v_3 \]

(a) $K_1$

(b) $K_2$

Note. $\sigma_L$ is time respecting while $\sigma_R$ not!
Temporal Motifs by Paranjape et al. (2017)

This model aims at providing a more general and flexible definition of temporal motifs.

**Data.** $G = (V, E), E = \{(u, v, t)\}$ be a directed temporal network

A temporal motif is a pair $M = (K, \sigma)$ (Liu et al., 2019) where

- $K$ is a directed and (weakly)connected multigraph with $k$-nodes and $\ell$-edges.

- $\sigma$ is an ordering of the edges of $K$ (modelling temporal dynamics of $K$)

**Example.** Fixing $K = K_1$ then

(a) $\sigma_L = \langle (v_1, v_2), (v_2, v_3) \rangle$

(b) $\sigma_R = \langle (v_2, v_3), (v_1, v_2) \rangle$

**Note.** $\sigma_L$ is time respecting while $\sigma_R$ not!
Temporal Motif Counting Problem (Paranjape et al., 2017)

Given $G$ and a value $\delta \in \mathbb{R}^+$, a time-ordered sequence $S = \langle (x'_1, y'_1, t'_1), \ldots, (x'_\ell, y'_\ell, t'_\ell) \rangle$ of $\ell$ unique edges from $G$ is a $\delta$-instance of $M = \langle (x_1, y_1), \ldots, (x_\ell, y_\ell) \rangle$ if

1. there exists a bijection $h$ from the vertices appearing in $S$ to the vertices of $M$, with $h(x'_i) = x_i$ and $h(y'_i) = y_i$, and $i \in [\ell]$;
2. the edges of $S$ occur within $\delta$ time; i.e., $t'_\ell - t'_1 \leq \delta$.

(a) Temporal graph $G$  (b) Temporal motif, $t_i$ gives ordering in $\sigma$
Temporal Motif Counting Problem (Paranjape et al., 2017)

Given $G$ and a value $\delta \in \mathbb{R}^+$, a time-ordered sequence $S = \langle (x'_1, y'_1, t'_1), \ldots, (x'_\ell, y'_\ell, t'_\ell) \rangle$ of $\ell$ unique edges from $G$ is a $\delta$-instance of $M = \langle (x_1, y_1), \ldots, (x_\ell, y_\ell) \rangle$ if

1. there exists a bijection $h$ from the vertices appearing in $S$ to the vertices of $M$, with $h(x'_i) = x_i$ and $h(y'_i) = y_i$, and $i \in [\ell]$;
2. the edges of $S$ occur within $\delta$ time; i.e., $t'_\ell - t'_1 \leq \delta$.

Fix $\delta = 10$

(a) Temporal graph $G$  (b) Temporal motif, $t_i$ gives ordering in $\sigma$
Temporal Motif Counting Problem (Paranjape et al., 2017)

Given $G$ and a value $\delta \in \mathbb{R}^+$, a *time-ordered* sequence $S = \langle (x'_1, y'_1, t'_1), \ldots, (x'_\ell, y'_\ell, t'_\ell) \rangle$ of $\ell$ unique edges from $G$ is a $\delta$-instance of $M = \langle (x_1, y_1), \ldots, (x_\ell, y_\ell) \rangle$ if

1. there exists a bijection $h$ from the vertices appearing in $S$ to the vertices of $M$, with $h(x'_i) = x_i$ and $h(y'_i) = y_i$, and $i \in [\ell]$;
2. the edges of $S$ occur within $\delta$ time; i.e., $t'_\ell - t'_1 \leq \delta$.

Fix $\delta = 10$

(a) Temporal graph $G$  
(b) Temporal motif, $t_i$ gives ordering in $\sigma$
Temporal Motif Counting Problem – cont.

Count of a temporal motif $M$ is: \# of $\delta$-instances of $M$ in $G$

**Problem**

Given a temporal network $G$, a temporal motif $M$ and a parameter $\delta \in \mathbb{R}^+$ obtain the count of the temporal motif $M$

The problem is **NP-hard**
Count of a temporal motif $M$ is: $\#$ of $\delta$-instances of $M$ in $G$

**Problem**

Given a temporal network $G$, a temporal motif $M$ and a parameter $\delta \in \mathbb{R}^+$ obtain the count of the temporal motif $M$

The problem is **NP-hard**

The problem is **NP-hard** even for motifs in $\textbf{P}$ for static networks (Liu et al., 2019)!

**Lets look at existing algorithms**
Exact Algorithm by Paranjape et al. (2017)

The proposed algorithm computes the counts of all \(\{2, 3\}\)-node 3-edge temporal motifs.
Exact Algorithm by Paranjape et al. (2017)

The proposed algorithm computes the counts of all \( \{2, 3\} \)-node 3-edge temporal motifs.

**General framework**

- Computes the aggregate graph \( G_A \) of \( G \)
- Enumerates all subgraphs \( H \subseteq G_A \) isomorphic to \( K \) (i.e., \( H \sim K \))
- For each \( H \) gathers the corresponding temporal networks \( G_H \) and sorts edges by timestamps
- Applies dynamic-programming to obtain the counts of all the sequences of length \( \ell \) in a window of size \( \delta \)
Exact Algorithm by Paranjape et al. (2017)

The proposed algorithm computes the counts of all \( \{2, 3\} \)-node 3-edge temporal motifs.

**General framework**

- computes the aggregate graph \( G_A \) of \( G \)
- enumerates all subgraphs \( H \subseteq G_A \) isomorphic to \( K \) (i.e., \( H \sim K \))
- for each \( H \) gathers the corresponding temporal networks \( G_H \) and sorts edges by timestamps
- applies dynamic-programming to obtain the counts of all the sequences of length \( \ell \) in a window of size \( \delta \)

**Runtime.** \( \mathcal{O}(|E| + n^k + \sum_{H \sim K} |E_{G_H}|(\log(|E_{G_H}|) + |E_H|^{\ell})) \)

Specialized routines for specific motif classes through dynamic programming
Other Exact Algorithms

*Other exact approaches*

(Mackey et al., 2018) enumerates all $\delta$-instances of a fixed temporal motif $M$ without constrains

(Pashanasangi and Seshadhri, 2021) Fast algorithms for temporal triangle counting based on degeneracy ordering

(Gao et al., 2022) improved algorithms for counting $\{2, 3\}$-node 3-edge temporal motifs

(Sarpe, 2023) improved (Mackey et al., 2018) by different matching criteria and timeline partition

(Yuan et al., 2023) dedicated hardware for counting temporal motifs

(Cai et al., 2023) exact algorithms for counting butterflies in temporal bipartite networks

... 

As the problem is hard, often better rely on *approximate counting*!
Given a temporal network $G$, a temporal motif $M$ and a parameter $\delta \in \mathbb{R}^+$, and two additional parameters $\varepsilon, \eta \in (0, 1)^2$ obtain $C'$ an estimate of the count $C$ of the temporal motif $M$ with

$$\mathbb{P}[|C' - C| \geq \varepsilon C] \leq \eta$$

Why approximate counts?
Motif Approximation Problem

Problem

Given a temporal network $G$, a temporal motif $M$ and a parameter $\delta \in \mathbb{R}^+$, and two additional parameters $\varepsilon, \eta \in (0, 1)^2$ obtain $C'$ an estimate of the count $C$ of the temporal motif $M$ with

$$\mathbb{P}[|C' - C| \geq \varepsilon C] \leq \eta$$

Why approximate counts?

- often efficient and practical to compute on massive data
- approximations are robust to noisy data
- guarantees on the quality of estimate
Approximation Algorithms

Most of approximate algorithm are based on randomized sampling

(Liu et al., 2019): partitioning time-span of $G$ in non-overlapping windows and uses importance sampling to decide windows to explore

(Wang et al., 2020): sampling temporal edges with fixed probability, specialized estimators for triangles, and streaming

(Sarpe and Vandin, 2021b; Sarpe, 2023): interval based algorithms performing uniform sampling without partitioning

(Pu et al., 2023): edges sampling techniques for counting temporal butterflies on undirected bipartite temporal networks

...
On the Selection of $\sigma$ (Sarpe and Vandin, 2021a)

A temporal motif is a pair $(K, \sigma)$, how to properly pick $\sigma$?

Multiple values of $\sigma$ may need to be tested!

Problem

Given a temporal network $G$, a parameter $\delta \in \mathbb{R}^+$, a static undirected subgraph $H$ and a value $\ell \geq |E_H|$

compute the count of all temporal motifs "mapping" on $H$ and having $\ell$ temporal edges

Example. Fix $\ell = 3$ then

(a) motifs to "count"
On the Selection of $\sigma$ (Sarpe and Vandin, 2021a)

A temporal motif is a pair $(K, \sigma)$, how to properly pick $\sigma$?

Multiple values of $\sigma$ may need to be tested!
On the Selection of $\sigma$ (Sarpe and Vandin, 2021a)

A temporal motif is a pair $(K, \sigma)$, how to properly pick $\sigma$?

Multiple values of $\sigma$ may need to be tested!

**Problem**

Given a temporal network $G$, a parameter $\delta \in \mathbb{R}^+$, a *static* undirected subgraph $H$ and a value $\ell \geq |E_H|$ compute the count of *all* temporal motifs “mapping” on $H$ and having $\ell$ temporal edges.

**Example.** Fix $\ell = 3$ then

(a) $H$

(b) motifs to “count”
Proposed algorithm \texttt{oden}:
randomized sampling algorithm + theoretical guarantees
**ODEN** (Sarpe and Vandin, 2021a)

Proposed algorithm **ODEN**: randomized sampling algorithm + theoretical guarantees

Efficiently estimates multiple temporal motifs counts simultaneously. $H$: triangle and data comes from Facebook posts, varying $ℓ$. 

![Graph showing distribution of motif counts with varying $ℓ$.](image)
Are there applications?
Are there applications?

Temporal motifs enabled both novel algorithmic problems and more nuanced applications.
Stochastic Block Models (Porter et al., 2022)

**Goal.** Obtain highly accurate stochastic block models (SBM) to capture temporal motif $\delta$-instances

**Proposed solution.** Temporal Activity SBM

1. partition nodes according to their activity level \{in, out\}-edges (resp. $C_{\text{in}}$, $C_{\text{out}}$ groups)
2. model temporal edges according

\[
\theta = \begin{pmatrix}
\theta_{1,1} & \theta_{1,2} \\
\theta_{2,1} & \theta_{2,2}
\end{pmatrix}
\]

where $\theta \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$ models edge occurrence
3. analytical computation of motif counts according to such model
Stochastic Block Models (Porter et al., 2022)

**Goal.** Obtain highly accurate stochastic block models (SBM) to capture temporal motif $\delta$-instances

**Proposed solution.** Temporal Activity SBM

1. partition nodes according to their activity level $\{\text{in, out}\}$-edges (resp. $C^{\text{in}}$, $c^{\text{out}}$ groups)
2. model temporal edges according

$$\theta = \begin{pmatrix}
\theta_{1,1} & \theta_{1,2} \\
\theta_{2,1} & \theta_{2,2}
\end{pmatrix}$$

where $\theta \in \mathbb{R}^{C^{\text{in}} \times C^{\text{out}}}$ models edge occurrence

3. analytical computation of motif counts according to such model

The model accurately tracks temporal motif counts. Financial dataset recorded over 10 years, $\delta = 90$ days, left $M_1 = \langle (v_1, v_2), (v_3, v_2), (v_1, v_2) \rangle$, right $M_2 = \langle (v_1, v_2), (v_2, v_1), (v_2, v_1) \rangle$
Synthetic Network Generators (Liu and Sarıyüce, 2023)

**Goal.** Obtain a synthetic temporal network, similar to the one in input for temporal motifs

**Proposed solution.** Motif Transition Model

Cold event (CE): first event on a temporal motif instance

1. compute temporal network statistics
   - static degree distribution ($K_{CE}$) and timestamps ($T_{CE}$) of cold events
   - $P$ motif-transition properties (how likely are motifs to evolve from one to another)
   - $\Lambda$ motif transition rated (how often they transition)
   - $\mu$ number of static edges involved in transitions.
2. generate static network from $K_{CE}$ and assign ($T_{CE}$)
3. simulate transition process according to computed metrics
Synthetic Network Generators (Liu and Sariyüce, 2023)

Goal. Obtain a synthetic temporal network, similar to the one in input for temporal motifs

Proposed solution. Motif Transition Model

Cold event (CE): first event on a temporal motif instance

1. compute temporal network statistics
   - static degree distribution ($K_{CE}$) and timestamps ($T_{CE}$) of cold events
   - $\mathcal{P}$ motif-transition properties (how likely are motifs to evolve from one to another)
   - $\Lambda$ motif transition rated (how often they transition)
   - $\mu$ number of static edges involved in transitions.

2. generate static network from $K_{CE}$ and assign ($T_{CE}$)

3. simulate transition process according to computed metrics
Mining Persistent Events (Belth et al., 2020)

**Goal.** Distinguish between how motif occur over time *streams*

\[
P(x) = \frac{W(x) \cdot F(x) \cdot S(x)}{	ext{width, frequency, uniformity}}
\]

Online and offline streaming algorithms are developed, efficient for small size of events (small \(\ell\)). The score allows to distinguish between frequent/infrequent and bursty/persistent.
Mining Persistent Events (Belth et al., 2020)

**Goal.** Distinguish between how motif occur over time *streams*

**Proposed solution.** Assign persistence score and algorithms to compute it

Let $x$ be an event (e.g., temporal motif instance) then the persistence $P(\cdot)$

$$P(x) = f \left( \frac{W(x)}{\text{width}}, \frac{F(x)}{\text{frequency}}, \frac{S(x)}{\text{uniformity}} \right)$$

*Online and offline* streaming algorithms are developed, efficient for small size of events (small $\ell$)

The score allows to distinguish between frequent/infrequence and bursty/persistent
Some Practical Applications (Liu et al., 2024)

- capturing high-order patterns for phishing gang identification on cryptocurrency networks
Some Practical Applications (Lei et al., 2020)

- analyzing different temporal travel patterns in people commuting (metro vs bike sharing)

(a) using metro

(b) using bikesharing
Other Temporal Motif Definitions

Several other definitions exist in literature

- (Boekhout et al., 2019): studied temporal multilayer motifs
- (Lee and Shin, 2023): studied temporal hypergraph motifs
- (Longa et al., 2021): studied motifs based ego-networks
- (Kosyfaki et al., 2018): defined motifs for temporal networks with flows

If you want to know more, check the survey by Liu et al. (2021)
STAY STRONG

IT'S ALMOST OVER
Diffusion and Random Networks
Diffusion Analysis and Spreading

- **propagation models**
  - used to study disease spreading or information cascade in the network
- **activity spreading**: virus, information, idea, rumor
- **applications**: epidemiology, information security, marketing
- **why use models?**
  - facilitate mathematical analysis of propagation processes
  - have intuitive interpretation
  - proven to be realistic by empirical studies
- **extensive survey in the book** (Shakarian et al., 2015)
Standard Models

Most used models are

- susceptible-infected (SI) model
  - SIR, SIRS, other variants
- independent cascade (IC) model
- linear threshold (LT) model

Such models are *important building blocks* for many *data mining formulations*!
Susceptible-Infectious (SI) Model

► beginning
  – time step $t_0$
  – one or several infected nodes in $I_{t_0}$ (seeds of infection)

► subsequent timestamp $t$
  – all infected nodes try to infect each of their susceptible neighbors
  – with probability $p$ infection is passed through an edge
  – if a node receives infection becomes infected

► the process continues until all nodes are infected.

Some other node types, recovered (nodes that were infectious and now cannot spread), and exposed (infected that cannot spread)
Independent Cascade (IC) Model

- nodes can be in either susceptible or infectious
- each edge \((u, v)\) has an individual infection probability (based on proximity, frequency, etc.)
- infected node \(u\) has a single chance to infect its neighbors

Used to study new propagation of ideas, concepts, or products (Kempe et al., 2003; Wang et al., 2012)
Linear Threshold (LT) Model

- every edge \((u, v)\) has a probability \(p(u, v)\)

- at the time step \(t\), \(u\): susceptible \(\rightarrow\) infectious, if the total weight from its infectious neighbors is larger than a random propagation threshold \(\theta_u\)

\[
\sum_{v \in N(u)} p(v, u) \mathbf{1}[v \text{ is infectious at } t] \geq \theta_u
\]

- conditional on thresholds and the initially infected nodes the process is *deterministic*.

LT model has applications in viral-marketing (Chen et al., 2010; Goyal et al., 2011)
Mining Applications

Powerful modeling for many mining primitives

1. **immunization strategies**, e.g., find smallest set of nodes to stop a spreading process. (Lee et al., 2012; Yu et al., 2010; Starnini et al., 2013; Génois et al., 2015; Mantzaris and Higham, 2016; Valdano et al., 2015; Gauvin et al., 2015)

2. **influence maximization**, e.g., select the initial set of seeds, to optimize diffusion, applications in marketing and network design. (Aggarwal et al., 2012; Zhuang et al., 2013; Gayraud et al., 2015; Rodriguez et al., 2011; Gomez-Rodriguez et al., 2016; Chen et al., 2012; Liu et al., 2012; Rodriguez and Schölkopf, 2012; Du et al., 2013)

3. **seed and cascade reconstruction**, e.g., given some observed data of a spreading phenomenon, find the most probable seed nodes or cascades, applications in epidemiology and influencer discovery. (Shah and Zaman, 2011; Lappas et al., 2010; Prakash et al., 2012)
Random Models

Common questions in temporal data analysis

- *how novel* is this result?
- is this only due to *random chance*?
- are there *properties in the data explaining* the results?

To find an *answer* → use a *statistical test* (Pellegrina et al., 2019)
Random Models

Common questions in temporal data analysis

- *how novel* is this result?
- is this only due to *random chance*?
- are there *properties in the data explaining* the results?

To find an *answer* → use a *statistical test* (Pellegrina et al., 2019)

- start from a *temporal network* \( G \)
- formulate an *hypothesis* \( (H_0) \) about data (e.g., time does not matter for \( f(G) \))
- perform a *test to reject* \( H_0 \), usually
  - generate multiple datasets \( G_1^{H_0}, \ldots, G_L^{H_0} \) for some large \( L \) according to \( H_0 \)
  - compute some function \( g(G_1^{H_0}, \ldots, G_L^{H_0}) \) to reject \( H_0 \) (e.g., \( g(\cdot) \) not explains \( f(G) \)).
Randomized models are used to test temporal/static properties in data
A temporal network as time-line of events

We have
1. static structure (SS)
2. timeline associated to its links (TL)

To obtain random models → use these two properties or combinations of the two
Randomized models are used to test temporal/static properties in data.
A temporal network as time-line of events.

We have
1. static structure (SS)
2. timeline associated to its links (TL)

To obtain random models → use these two properties or combinations of the two

Let us see some examples.
Shuffling *only* static properties while fixing the temporal ones

(a) *link shuffling*

(b) *(conn.) constrained degree link shuffling*
Random Models Gauvin et al. (2022)

timeline shuffling

This model retains static properties and conditions on the observed temporal ones
Other models retaining static properties

(a) *shuffling events over each timeline*

(b) *shuffling events and retaining gaps*

... and much more such combinations (static + temporal) ...
Random Models Gauvin et al. (2022)

Some random models for snapshot-based temporal networks

(a) *snapshot shuffling*

(b) *isomorphism based*
Random Models

Summary.

- *random models* can be of **fundamental importance** for testing significance/generating additional data
- they can be **applied for most of the mining problems that we discussed**
- some of them may be *hard to compute* and **new methods may be required**
Agenda

Part I: Introduction and Motivation
  ▶ models of temporal networks
  ▶ algorithmic approaches

Part II: Mining Temporal Networks A:
  ▶ connectivity, temporal properties
  ▶ centrality, cores

Part III: Mining Temporal Networks B:
  ▶ communities, patterns and events
  ▶ diffusion and random networks

Part IV: Tools and Code Libraries

Part V: Challenges, Open Problems, and Trends
Part IV

Tools and Code Libraries
<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Language</th>
<th>Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNAP</td>
<td>C++/Python</td>
<td>Temporal motifs</td>
</tr>
<tr>
<td>Graph-tool</td>
<td>Python/C++</td>
<td>Simulate network dynamics (e.g., spreading)</td>
</tr>
<tr>
<td>Teneto</td>
<td>Python</td>
<td>Temporal network measures (centrality, reachability, etc..), community detection, visualization</td>
</tr>
<tr>
<td>Phasik</td>
<td>Python</td>
<td>Infer temporal networks from time series data</td>
</tr>
<tr>
<td>Reticula</td>
<td>C++/Python</td>
<td>Random networks, random models, temporal reachability, events...</td>
</tr>
<tr>
<td>Tglib</td>
<td>C++/Python</td>
<td>Paths, centrality and other properties (cores, clustering coefficient, etc..)</td>
</tr>
<tr>
<td>Raphtory</td>
<td>Rust/Python</td>
<td>Centrality, communities, cores, motifs, null models, visualization and more...</td>
</tr>
</tbody>
</table>
Agenda

Part I: Introduction and Motivation
- models of temporal networks
- algorithmic approaches

Part II: Mining Temporal Networks A:
- connectivity, temporal properties
- centrality, cores

Part III: Mining Temporal Networks B:
- communities, patterns and events
- diffusion and random networks

Part IV: Tools and Code Libraries

Part V: Challenges, Open Problems, and Trends
Part V

Challenges, Open Problems, and Trends
Challenges in Temporal Network Mining

- large number of **problem formulations** and variants
- gaps **fundamental theoretical** treatment
  - many are **combinations of several** ideas of static cases
  - require often **many** parameters
- hard to compare methods and **choose based on applications**
  - **few datasets** with ground-truth solutions
  - synthetic generators are built on **various assumptions**
  - **no** standards and benchmarks
  - as always: lack of useful and rich datasets
- a large number of **quality metrics** to calculate and compare
- comparisons are **misleading** if methods are designed for other definitions
Directions in Temporal Network Mining

- **more systematic** approaches, **quality guarantees**
- **interpretability** of the results
- **diversity** and **fairness**

- **applications** and **application-tailored algorithms**
  - encourage *interdisciplinary research* and collaborations
  - computational *social science*
Thanks for your attention!

https://miningtemporalnetworks.github.io/menti.com

code: 14 46 97 6


references XIII


