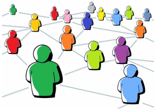
Data analysis – What data and what to look for?

Input: Real-world data from mobile networks (e.g. CRAWDAD datasets)

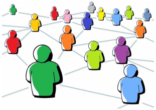


Question: Can we detect basic temporal features?

Data analysis – What data and what to look for?

Input: Real-world data from mobile networks (e.g. CRAWDAD datasets)

Question: Can we detect basic temporal features?



Examples of properties (seen previously):

1. $\mathcal{J}^{1\forall}$: At least one node can reach all the others through a journey ($1 \rightsquigarrow *$),
2. $\mathcal{TC}$: All nodes can reach each other through journeys ($* \rightsquigarrow *$),
3. $\mathcal{E}^{1\forall}$: At least one node shares at some point an edge with every other ($1 - *$),
4. $\mathcal{K}$: All pairs of nodes share at some point an edge ($* - *$),
5. $\mathcal{J}^{\forall 1}$: At least one node can be reached by all others through a journey ($* \rightsquigarrow 1$),
6. $\mathcal{J}^{1\forall >}$: At least one node can reach all the others through a strict journey ($1 \overset{st}{\rightsquigarrow} *$),
7. $\mathcal{TC}^{>}$: All nodes can reach each other through strict journeys ($* \overset{st}{\rightsquigarrow} *$).

Data analysis – Defining intermediate objects

Input: $\mathcal{G} = \{G_1, ..., G_k\}$, with $G_i = (V, E_i)$.

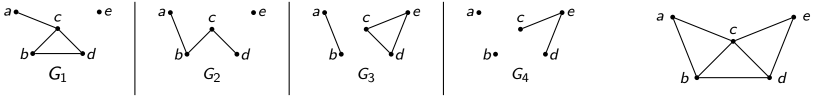
Data analysis – Defining intermediate objects

Input: $\mathcal{G} = \{G_1, ..., G_k\}$, with $G_i = (V, E_i)$.

(1) The footprint

The *footprint* of $\mathcal{G}$ is the (static) graph $G = (V, E)$ such that $E = \cup_i E_i$. Here, we can assume that V_i does not vary.

For example: this temporal graph... ... has footprint



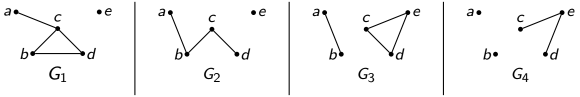
Data analysis – Defining intermediate objects

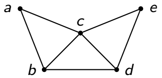
Input: $\mathcal{G} = \{G_1, ..., G_k\}$, with $G_i = (V, E_i)$.

(1) The footprint

The *footprint* of $\mathcal{G}$ is the (static) graph $G = (V, E)$ such that $E = \cup_i E_i$. Here, we can assume that V_i does not vary.

For example: this temporal graph... ... has footprint

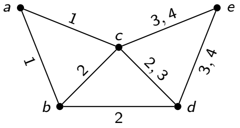


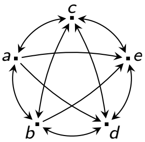


(2) The transitive closure of journeys

The *transitive closure* of the journeys is the <u>directed</u> graph $\overrightarrow{G} = (V, \overrightarrow{E})$ such that $(u, v) \in \overrightarrow{E}$ if and only if there is a journey from u to v in $\mathcal{G}$. We distinguish between *strict* and *non-strict* transitive closures, depending on the type of journeys allowed.

For example, this temporal graph... ...has strict transitive closure





Data analysis – Using the intermediate objects

Now, we have the following equivalences:

- $\mathcal{J}^{1\forall} \iff$ The transitive closure contains an out-dominating set of size 1.
- $\mathcal{J}^{\forall 1} \iff$ The transitive closure contains an in-dominating set of size 1.
- $\mathcal{J}^{1\forall >} \iff$ The strict transitive closure contains an out-dominating set of size 1.
- $\mathcal{TC} \iff$ The transitive closure is a complete graph.
- $\mathcal{TC}^{>} \iff$ The strict transitive closure is a complete graph.
- $\mathcal{E}^{1\forall} \iff$ The footprint contains a dominating set of size 1 (*a.k.a.* universal vertex).
- $\mathcal{K} \iff$ The footprint is a complete graph.

Data analysis – Using the intermediate objects

Now, we have the following equivalences:

- $\mathcal{J}^{1\forall} \iff$ The transitive closure contains an out-dominating set of size 1.
- $\mathcal{J}^{\forall 1} \iff$ The transitive closure contains an in-dominating set of size 1.
- $\mathcal{J}^{1\forall >} \iff$ The strict transitive closure contains an out-dominating set of size 1.
- $\mathcal{TC} \iff$ The transitive closure is a complete graph.
- $\mathcal{TC}^{>} \iff$ The strict transitive closure is a complete graph.
- $\mathcal{E}^{1\forall} \iff$ The footprint contains a dominating set of size 1 (*a.k.a.* universal vertex).
- $\mathcal{K} \iff$ The footprint is a complete graph.

$\rightarrow$ Queries can be answered trivially once the intermediate objects are computed!

Exercises: how to compute these objects?

Solutions to the exercises

In all algorithms, the input $\mathcal{G}$ is given as a sequence $\{G_1, ..., G_k\}$, with $G_i = (V, E_i)$.

Algorithm 1 Computing the footprint

```
E ← ∅ // edges of the footprint
for all G_i ∈ 𝒢 do
    for all e ∈ E_i do
        E ← E ∪ e
return (V, E) // the footprint graph
```

Exercises: how to compute these objects?

Solutions to the exercises

In all algorithms, the input $\mathcal{G}$ is given as a sequence $\{G_1, ..., G_k\}$, with $G_i = (V, E_i)$.

Algorithm 3 Computing the footprint

$E \leftarrow \emptyset$ // edges of the footprint
for all $G_i \in \mathcal{G}$ **do**
  for all $e \in E_i$ **do**
    $E \leftarrow E \cup e$
return (V, E) // the footprint graph

Algorithm 4 Computing the <u>strict</u> transitive closure

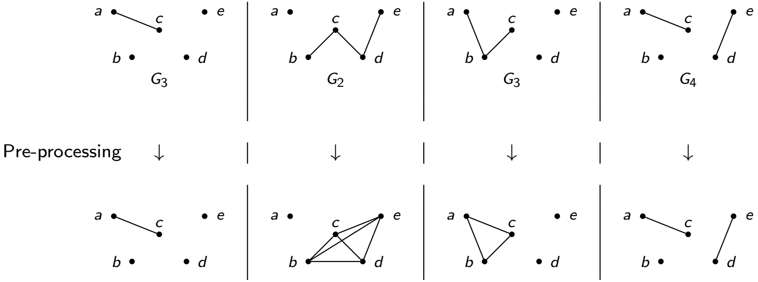
$T \leftarrow (V, \emptyset)$ // initial transitive closure
for all $G_i \in \mathcal{G}$ **do**
  $T_{new} \leftarrow copy(T)$
  for all $(u, v) \in E_i$ // manipulated as a directed graph **do**
    for all w such that $(w, u) \in E(T)$ **do**
      $E(T_{new}) \leftarrow E(T_{new}) \cup (w, v)$
    $E(T_{new}) \leftarrow E(T_{new}) \cup (u, v)$
return T

Solutions to exercises (2)

Algorithm 3 Computing the transitive closure for non-strict journeys

$\rightarrow$ 2 steps:

1. Pre-process each graph of the sequence by saturating the connected components



2. Then runs the previous algorithm for *strict* transitive closure on this new sequence.

$\rightarrow$ There is an arc in the result if and only if there is a (possibly non-strict) journey in $\mathcal{G}$.