Algorithmic Aspects of Temporal Betweenness

Sebastian Buß   Hendrik Molter   Rolf Niedermeier   Maciej Rymar

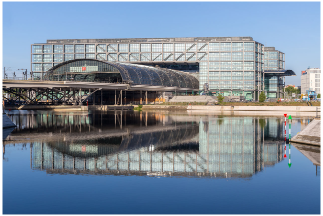
Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Germany

Algorithmic Aspects of Temporal Graphs III

Betweenness Centrality: Motivation I

“How important is Berlin main station as a hub for the public transportation network?”

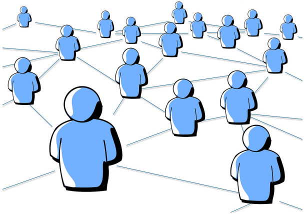
Betweenness Centrality: Motivation II



Transportation Networks

Routing Networks

Protein Networks



Social Networks

Betweenness Centrality: Definition

Betweenness of a vertex v in a graph $G = (V, E)$:

"How likely is a shortest path to pass through vertex v?"

Betweenness Centrality

$$C_B(v) = \sum_{s \neq v \neq z} \frac{\sigma_{sz}(v)}{\sigma_{sz}}$$

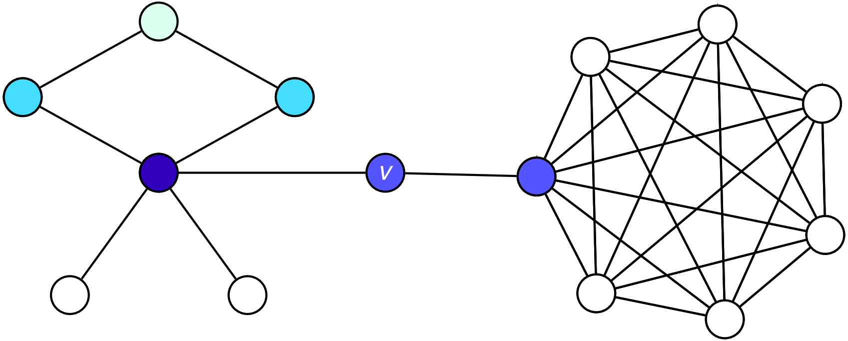
σ_{sz}: # shortest paths from s to z $\quad$ $\sigma_{sz}(v)$: # shortest paths from s to z via v

A remark on motivation:

Betweenness assumes information travels along optimal paths!

In many scenarious unrealistic $\rightarrow$ Random walk based centralities (e.g. PageRank).

Betweenness Centrality: Example



Darker colors indicate higher betweenness centrality.

$$C_B(v) = \sum_{s \neq v \neq z} \frac{\sigma_{sz}(v)}{\sigma_{sz}} = 42.$$

Betweenness Centrality: Background

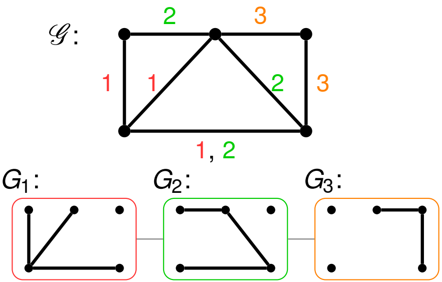
- First formally described by Linton Freeman in 1977
- Freeman was Prof. for Sociology at UC Irvine and founder of the journal “Social Networks”
- Measure for quantifying the control on the communication in a social network

- Ulrik Brandes, Prof. for Social Networks at ETH Zürich
- Published “blueprint” for all modern betweenness algorithms in 2001 (J. Math. Sociol.): Brandes’ algorithm
- Main achievement: improved running time for sparse graphs, linear space requirement

Temporal Graphs and Temporal Paths: Definition

Temporal Graph

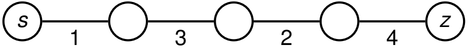
A **temporal graph** $\mathcal{G} = (V, (E_i)_{i\in[\tau]})$ is a vertex set V with a list of edge sets $E_1, \ldots, E_\tau$ over V, where τ is the lifetime of $\mathcal{G}$.



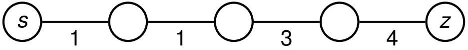
Temporal (s, z)-Path

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

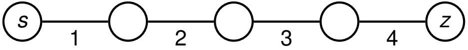
- increasing time stamps (strict).
- non-decreasing time stamps (non-strict).



Not a temporal path.



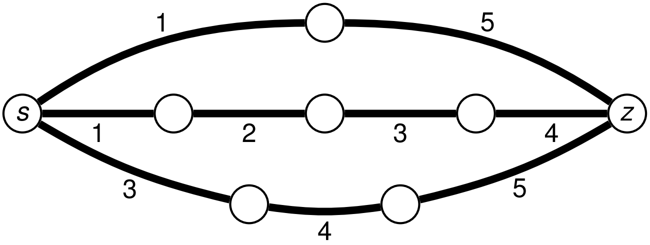
Non-strict temporal path. (Not strict.)



Temporal path (both strict and non-strict).

Optimal Temporal Paths

Method: Replace **shortest path** by **"optimal" temporal path**.
Problem: Which temporal path from s to z is optimal?



- **Shortest** temporal paths use the minimum number of edges.
- **Foremost** temporal paths have a minimum arrival time.
- **Fastest** temporal paths have a minimum difference between starting and arrival time.

Most well-motivated: **Foremost** temporal paths.

Temporal Betweenness Centrality

Temporal Betweenness Variants:

(strict vs. non-strict) $\times$ (# optimality criteria) $=$
six temporal betweenness variants

$\{$**Strict, Non-Strict**$\}$ $\{$**Shortest, Foremost, Fastest**$\}$ **Temporal Betweeness**

What is known:

- Temporal Betweenness has been already been studied extensively.
- Many more combinations and considerations are possible and have been made.
- Most approaches use static expansions and use known algorithms for static betweenness.

Question: Which variants are computable with a “temporal version” of Brandes’ algorithm?

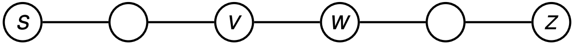
Main Ideas of Brandes' Algorithm

Recall: $C_B(v) = \sum_{s \neq v \neq t} \sigma_{st}(v)/\sigma_{st}$

Main Idea: Cleverly sum up "dependencies" $\delta_{s\bullet}(v)$ in a modified BFS without calculating them explicitly.

$$C_B(v) = \sum_{s \in V \setminus \{v\}} \delta_{s\bullet}(v), \text{ where } \delta_{s\bullet}(v) = \sum_{t \in V \setminus \{v\}} \sigma_{st}(v)/\sigma_{st}.$$

Use a recursive formula for $\delta_{s\bullet}(v)$ based on "successors" in shortest paths starting at s.



Vertex w is a "successor" of v (with respect to s).

(Presumably) Necessary conditions for this approach:

- Counting shortest/optimal paths is easy.
- "Successor relation" is acyclic.

Obstacles for Tractability I: Hard Counting

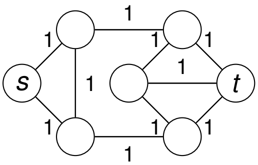
Observation

Computing betweenness values is at least as hard as counting optimal paths.

Static case:

Known facts

- Counting shortest paths from s to z can be done in poly time.
- Counting **all** paths from s to z is #P-hard [Valiant 79].

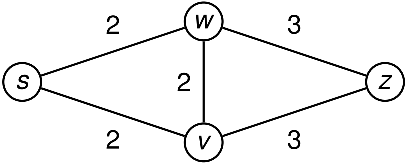


Corollary

Counting non-strict **foremost** or **fastest** temporal paths is #P-hard.

Obstacles for Tractability II: Cyclic Successors

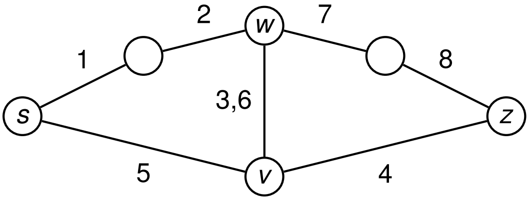
Foremost / Fastest temporal paths:



Observations:

- w is a successor of v
- v is a successor of w

Shortest Temporal Paths:

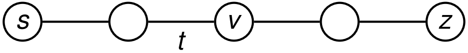


- w is a successor of v
- v is a successor of w

Cyclic Successors: Solution

Main idea: Consider "vertex appearances" instead of vertices.

Define: Vertex appearance (v, t) is "visited" by a temporal path if the path **arrives** in vertex v at time t.



Temporal Betweenness Centrality of Vertex Appearances

$$C_B^{(\star)}(v,t) = \sum_{s \neq v \neq z} \delta_{sz}^{(\star)}(v,t), \text{ where } \delta_{sz}^{(\star)}(v,t) = \begin{cases} 0 & \nexists \text{ temp.}(s,z)\text{-path} \\ \frac{\sigma_{sz}^{(\star)}(v,t)}{\sigma_{sz}^{(\star)}} & \text{otherwise} \end{cases}$$

$\sigma_{sz}^{(\star)}$: # $\star$-temp. paths from s to z $\sigma_{sz}^{(\star)}(v,t)$: # $\star$-temp. paths from s to z via (v,t)

$\star \in \{\text{foremost, fastest, shortest}\}$

Brandes Algorithm for Temporal Betweenness

Main recipe to transfer Brandes algorithm to the temporal setting:

- Define dependencies in an analogous way.
- Show a recursive formula for the dependencies similar to the static case.

Main differences to the static setting:

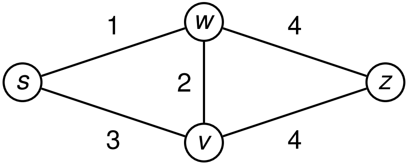
- Counting optimal paths can be #P-hard in the temporal setting (foremost & fastest).
  $\rightsquigarrow$ Presumably impossible to adapt Brandes for these optimality criteria.
- Successors can behave “cyclicly”.
  $\rightsquigarrow$ Necessary to consider vertex appearances.
- Walks can be optimal.

Temporal Betweenness: Tractable Variants for Foremost Temp. Paths

Shortest Foremost Temporal Paths

Shortest temporal paths among all foremost temporal paths.

Techniques for shortest temporal paths directly adaptable.



Strict **Prefix-Foremost** Temporal Paths

Foremost temporal paths where every **prefix** is also a foremost temporal paths.

Lemma [Wu et al. 16]

A prefix foremost temporal path always exists.

Techniques for shortest temporal paths adaptable with some modifications.

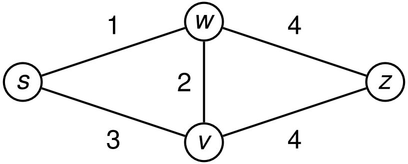
Temporal Betweenness: Prefix Foremost Temporal Paths

Observation

Counting **non-strict** prefix foremost temporal paths is #P-hard.

Observation

Time steps at which vertices are visited by prefix foremost temporal paths (starting from s) are unique.



For temporal betweenness based on strict prefix foremost temporal paths, we can consider vertices (instead of vertex appearances). $\rightsquigarrow$ Faster and more space efficient algorithm.

Our Results

Table of theoretical results:

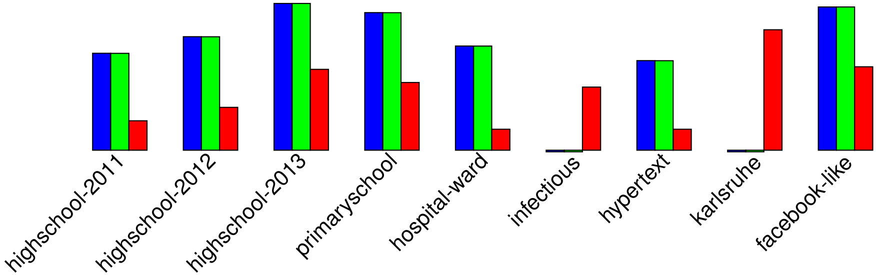
	strict	non-strict
Shortest	$O(n^3 \cdot \tau^2)$ time, $O(n \cdot \tau + M)$ space	
Foremost	#P-hard	
Fastest	#P-hard	
Prefix-foremost	$O(n \cdot M \cdot \log M)$ time, $O(n + M)$ space	#P-hard
Shortest foremost	$O(n^3 \cdot \tau^2)$ time, $O(n \cdot \tau + M)$ space	

n: # vertices M: # time edges τ: lifetime

Main Messages from Empirical Evaluation:

- Two prefix-foremost and shortest foremost variants produce similar vertex rankings based on betweenness score.
- Prefix-foremost betweenness can be computed much faster.

Temporal Betweenness Experiments I: Running Time



- Blue: non-strict shortest (foremost) betweenness.
- Green: strict shortest (foremost) betweenness.
- Red: strict prefix foremost betweenness.
- Time is on a log-scale, ranges from few minutes up to three hours.
- “infectious” and “karlsruhe” not solved for (non-)strict shortest (foremost) betweenness within three hours.

Temporal Betweenness Experiments II: Top 10 Vertices

Strict vs. non-strict

Data Set	shortest	sh. foremost
highschool-2011	10	8
highschool-2012	10	9
highschool-2013	10	8
primaryschool	9	9
hospital-ward	10	9
hypertext	10	10
facebook-like	10	10

Comparison of strict variants

Data Set	sh vs. sh fm	sh vs. p fm	sh fm vs. p fm
highschool-2011	5	4	9
highschool-2012	4	3	8
highschool-2013	4	3	7
primaryschool	0	0	8
hospital-ward	7	7	9
hypertext	4	4	9
facebook-like	9	6	7

Insights:

- Strict vs. non-strict makes not much difference.
- Shortest behaves quite differently to foremost variants.
- Foremost variants are very similar.

Conclusion

Summary:

- Several betweenness variants in the temporal setting.
- Foremost & fastest are #P-hard.
- Shortest, shortest foremost, and strict prefix foremost polynomial-time computable.

Insights from Experiments:

- Strict vs. non-strict makes not much difference.
- Shortest behaves quite differently to foremost variants while foremost variants are very similar.
- Strict prefix foremost is fastest to compute.

Link to arXiv.

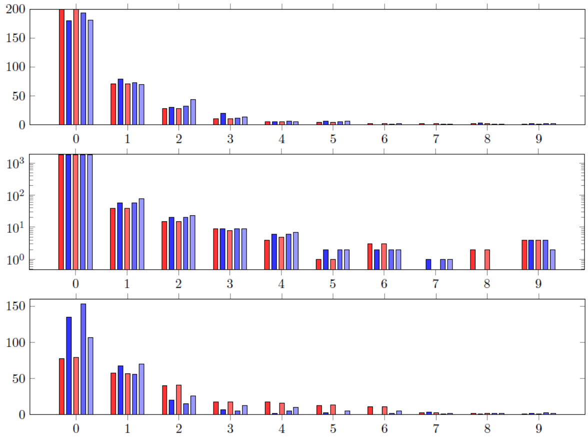
Thank you!

Temporal Betweenness Experiments Backup I: Running Time

Data Set	# Vtc's n	# Edges M	Lifetime T	NStr. Sh (Fm)	Str. Sh (Fm)	Str. P Fm
highschool-2011	126	28,560	272,330	$6.08 \cdot 10^1$	$6.05 \cdot 10^1$	$7.04 \cdot 10^{-1}$
highschool-2012	180	45,047	729,500	$1.82 \cdot 10^2$	$1.81 \cdot 10^2$	$1.74 \cdot 10^0$
highschool-2013	327	188,508	363,560	$2.44 \cdot 10^3$	$2.43 \cdot 10^3$	$2.1 \cdot 10^1$
primaryschool	242	125,773	116,900	$8.94 \cdot 10^2$	$8.89 \cdot 10^2$	$8.86 \cdot 10^0$
hospital-ward	75	32,424	347,500	$9.82 \cdot 10^1$	$9.79 \cdot 10^1$	$4.31 \cdot 10^{-1}$
infectious	10,972	415,912	6,946,340	-1	-1	$6.52 \cdot 10^0$
hypertext	113	20,818	212,340	$3.74 \cdot 10^1$	$3.72 \cdot 10^1$	$4.65 \cdot 10^{-1}$
karlsruhe	1,870	461,661	123,837,267	-1	-1	$2.88 \cdot 10^2$
facebook-like	1,899	59,835	16,736,181	$1.3 \cdot 10^3$	$1.3 \cdot 10^3$	$2.49 \cdot 10^1$

Running time given in seconds, a -1 indicates that the instance was not solved within three hours.

Temporal Betweenness Experiments Backup II: Betweenness Histograms



Histogram of betweenness values

- Data sets “highschool-2013”, “facebook-like”, “primaryschool” (top to bottom).
- Vertices are collected in 10 evenly distributed buckets between 0 and the highest temporal betweenness value.
- The temporal betweenness types from left to right: non-strict shortest, non-strict shortest foremost, strict shortest, strict shortest foremost, strict prefix foremost.

  Red-ish: shortest variants.

  Blue-ish: foremost variants.