Beyond Static Graphs

Algorithms for Temporal Network Analysis

Lutz Oettershagen

University of Liverpool

NetworKit Day 2026

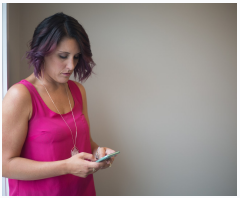
https://lutzoe.github.io/



Interactions come with times



Transportation
Trips and transfers



Communication
Messages and calls



Contact networks
Meetings and encounters

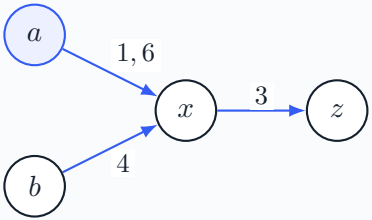


Infrastructure
Services and data flows

Each record says who interacted with whom, and when.

A temporal network is an event record

A temporal network records who interacts with whom, and when.

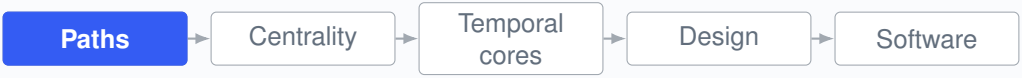


Directed event	Departure
$a \to x$	1
$x \to z$	3
$b \to x$	4
$a \to x$	6

The aggregate graph keeps the edges and discards their event times.

One event record, four questions

1	**What can reach what, and when?**	Temporal paths: scan events in time order.
2	**Which nodes matter?**	Centrality: count time-respecting paths and walks.
3	**Which groups interact closely in time?**	Temporal cores: peel edges in time windows.
4	**How should we change the schedule?**	Design: choose event times under a budget.



What can reach what, and when?

A temporal graph model

$$\mathcal{G} = (V, \mathcal{E}), \qquad e = (u, v, t, \lambda)$$

u, v	tail and head
t	departure time
$\lambda > 0$	traversal time
$t + \lambda$	arrival time

Unless stated otherwise: directed events, $\lambda = 1$, unrestricted waiting.

Temporal walks and paths

Temporal walk

Events $e_i = (u_i, v_i, t_i, \lambda_i)$ with

$$v_i = u_{i+1}, \qquad t_i + \lambda_i \leq t_{i+1}.$$

Temporal path

A walk that visits every vertex at most once.

Walk: a, x, b, x, z

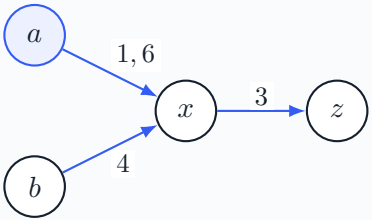


The same vertex x occurs twice.

Path: a, x, z



The event order determines usable routes



Starting at a at time 0
Depart for x at 1, arrive at 2.
Catch $x \to z$ at 3, arrive at 4.

Starting at b at time 0
Depart for x at 4, arrive at 5.
The departure to z at 3 has passed.

Both sources have an aggregate route to z, but only a can use it.

Temporal graphs are not dynamic graphs

Replay the record as edge insertions.
Does b reach z after each update?

t	Update	Dynamic	Temporal
1	insert $a \to x$	no	no
3	insert $x \to z$	no	no
4	insert $b \to x$	**yes**	**no**
6	insert $a \to x$	**yes**	**no**

Dynamic graph
An edge exists from insertion until deletion. Questions concern the *current* graph.

Temporal graph
An event is an opportunity at one time. Questions concern time-respecting walks over the *whole history*.

Consequence: temporal reachability is not transitive. b reaches x and x reaches z, yet b does not reach z.

Complementary: updating temporal answers as new events arrive is a dynamic problem.[1],[2]

[1] **O.** and Mutzel. SDM, 2023. [2] Béres et al. Applied Network Science, 2018.

Earliest, fastest, and shortest can disagree

Journey from s to z	Departure	Arrival	Duration	Hops
P_1	1	**5**	4	2
P_2	6	9	**3**	3
P_3	2	10	8	**1**

Each objective selects a different journey:[3],[4]

- **Earliest arrival** minimizes the arrival time.
- **Fastest** minimizes arrival minus departure.
- **Shortest** minimizes the number of events.

[3] Bui-Xuan et al. IJFCS, 2003. [4] Wu et al. PVLDB, 2014.

Earliest arrival via one chronological scan

Given source s, availability t_0, and deadline T, compute arrival labels $A[v]$.

One arrival label per vertex

$A[s] \leftarrow t_0, \quad A[v] \leftarrow \infty$ for $v \neq s$.

Scan events (u, v, t, λ) in departure order:[4]

if $A[u] \leq t$ and $t + \lambda \leq T$:

$A[v] \leftarrow \min\{A[v],\ t + \lambda\}$

Why one label suffices

An earlier arrival can wait for every departure a later arrival can use.

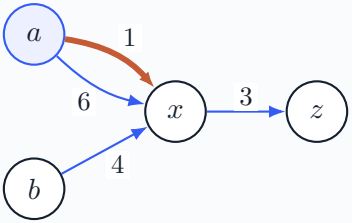
Positive traversal times put every usable predecessor earlier in the stream.

Reachability: v is reachable from (s, t_0) by T exactly when $A[v] < \infty$.

[4] Wu et al. PVLDB, 2014.

Earliest arrival on the running example

Source $s = a$, $t_0 = 0$, $T = 8$. Initially $A[a] = 0$, all other labels ∞.

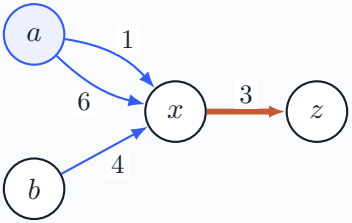


Event	$A[x]$	$A[z]$	$A[b]$	Decision
$(a, x, 1, 1)$	2	∞	∞	Reach x at 2

[4] Wu et al. PVLDB, 2014.

Earliest arrival on the running example

Source $s = a$, $t_0 = 0$, $T = 8$. Initially $A[a] = 0$, all other labels ∞.

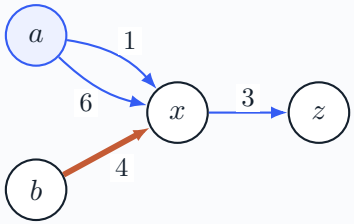


Event	$A[x]$	$A[z]$	$A[b]$	Decision
$(a, x, 1, 1)$	2	∞	∞	Reach x at 2
$(x, z, 3, 1)$	2	4	∞	Wait, reach z at 4

[4] Wu et al. PVLDB, 2014.

Earliest arrival on the running example

Source $s = a$, $t_0 = 0$, $T = 8$. Initially $A[a] = 0$, all other labels ∞.

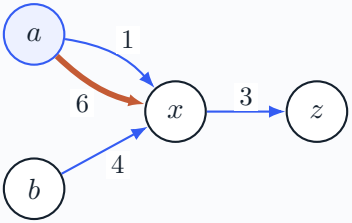


Event	$A[x]$	$A[z]$	$A[b]$	Decision
$(a, x, 1, 1)$	2	∞	∞	Reach x at 2
$(x, z, 3, 1)$	2	4	∞	Wait, reach z at 4
$(b, x, 4, 1)$	2	4	∞	Skip: $A[b] = \infty$

[4] Wu et al. PVLDB, 2014.

Earliest arrival on the running example

Source $s = a$, $t_0 = 0$, $T = 8$. Initially $A[a] = 0$, all other labels ∞.

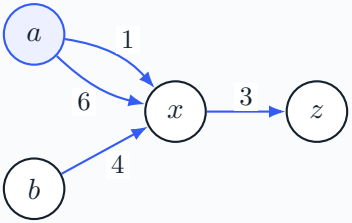


Event	$A[x]$	$A[z]$	$A[b]$	Decision
$(a, x, 1, 1)$	2	∞	∞	Reach x at 2
$(x, z, 3, 1)$	2	4	∞	Wait, reach z at 4
$(b, x, 4, 1)$	2	4	∞	Skip: $A[b] = \infty$
$(a, x, 6, 1)$	2	4	∞	7 does not improve 2

[4] Wu et al. PVLDB, 2014.

Earliest arrival on the running example

Source $s = a$, $t_0 = 0$, $T = 8$. Initially $A[a] = 0$, all other labels ∞.



Event	$A[x]$	$A[z]$	$A[b]$	Decision
$(a, x, 1, 1)$	2	∞	∞	Reach x at 2
$(x, z, 3, 1)$	2	4	∞	Wait, reach z at 4
$(b, x, 4, 1)$	2	4	∞	Skip: $A[b] = \infty$
$(a, x, 6, 1)$	2	4	∞	7 does not improve 2

Cost: $O(n + m)$ time and $O(n)$ extra memory on a time-sorted stream.[4]

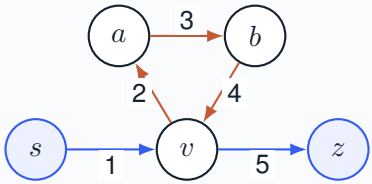
Sorting integer timestamps takes $O(m)$ with radix sort; logs often arrive sorted anyway.

[4] Wu et al. PVLDB, 2014.

Bounding waiting time makes paths NP-hard

A **Δ-restless path** repeats no vertex and waits at most Δ at every intermediate vertex.
Revisits can bridge a long wait.
With $\Delta = 1$ and instantaneous contacts:

$$s \xrightarrow{1} v \xrightarrow{2} a \xrightarrow{3} b \xrightarrow{4} v \xrightarrow{5} z$$



The walk is feasible. The simple route $s \to v \to z$ waits $4 > 1$.

- Restless *path* existence is NP-complete, even for $\Delta = 1$ and three timestamps.[5]
- Restless *walk* reachability is polynomial.[6],[7]

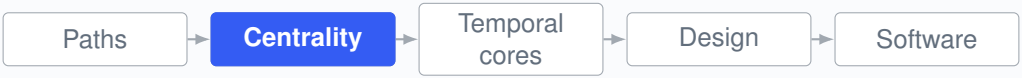
[5] Casteigts et al. Algorithmica, 2021. [6] Bentert et al. Applied Network Science, 2020. [7] Brunelli and Viennot. SAND, 2023.

Tractability depends on objective, waiting, and simplicity

Problem	State or constraint	Complexity
Earliest arrival	One arrival label per vertex	$O(n+m)$ on a sorted stream[4]
Fastest	Departure–arrival pairs	Polynomial[4]
Restless walks	Bounded waits, revisits allowed	Polynomial[6],[7]
Restless paths	Bounded waits, no revisits	NP-hard[5]

Reachable sets and arrival times are the raw material for the next two sections.

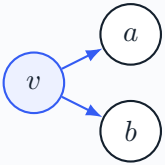
[4] Wu et al. PVLDB, 2014. [6] Bentert et al. Applied Network Science, 2020. [7] Brunelli and Viennot. SAND, 2023.
[5] Casteigts et al. Algorithmica, 2021.



Which nodes matter?

Centrality for identifying important nodes

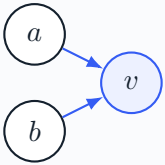
Source



Many chances to send onward.

Use: dissemination.

Receiver



Many chances to obtain.

Use: monitoring.

Relay



Receives, then passes on.

Use: intervention.

Timestamps can reverse a ranking

Score $R(v)$: number of other vertices reachable from v.[8]

Running record and a timestamp swap

Event	Record I	Record II
$a \to x$	1, 6	4, 6
$b \to x$	4	1
$x \to z$	3	3

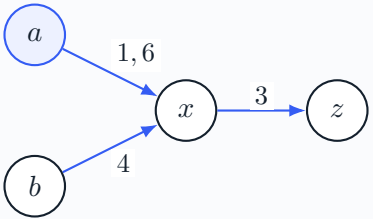
Reachability scores

Vertex	Record I	Record II
a	**2**	1
b	1	**2**
x	1	1
z	0	0

The aggregate graph, and every static score, is identical for both records.

[8] Pan and Saramäki. Physical Review E, 2011.

Relay scores: a relay must receive before it sends



Which walks can x pass on to z?

Arrival at x	From	Leaves x later?
2	a, departs 1	✓ via $x \to z$ at 3
5	b, departs 4	×
7	a, departs 6	×

Aggregate graph: the walks $a \to x \to z$ and $b \to x \to z$ both pass through x.

Time-respecting: only a's walk does. A relay score should count exactly these walks.

Fast temporal centrality: prune, index, stream

Top-k temporal closeness[9]
Bound each source's score and stop early. Up to $26\times$ faster on eight real networks.

Closeness index for evolving graphs[1]
Precompute an index for fast single-source temporal distance queries.

Temporal walk centrality[10]
Count every time-respecting walk through a node, weighted by waiting time. Two streaming passes, $O(|\mathcal{E}| \cdot \tau_{\max})$.

Temporal H-index[11]
A recursive local score in one streaming pass. Tracks simulated spreading and defines *pseudocores*.

Common theme: never enumerate paths; reuse one pass over the time-ordered events.

[9] **O.** and Mutzel. KAIS, 2022. [1] **O.** and Mutzel. SDM, 2023. [10] **O.** et al. WWW, 2022. [11] **O.** et al. KDD, 2023.



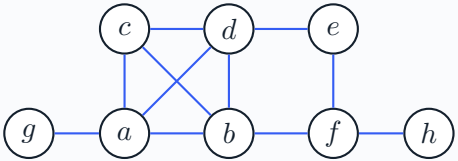
Which groups interact closely in time?

The static k-core: peel low-degree vertices

k-core: the maximal subgraph in which every vertex has at least k neighbors *inside* the subgraph.

Core number of v: the largest k such that v is in the k-core.

Peeling: repeatedly remove a vertex of minimum degree. Its degree at removal (never below earlier values) is its core number.



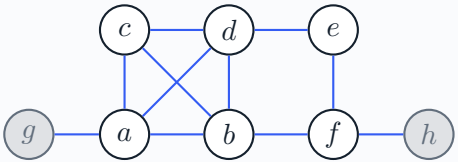
[12] Batagelj and Zaveršnik. arXiv, 2003.

The static k-core: peel low-degree vertices

k-core: the maximal subgraph in which every vertex has at least k neighbors *inside* the subgraph.

Core number of v: the largest k such that v is in the k-core.

Peeling: repeatedly remove a vertex of minimum degree. Its degree at removal (never below earlier values) is its core number.



core 1: g, h

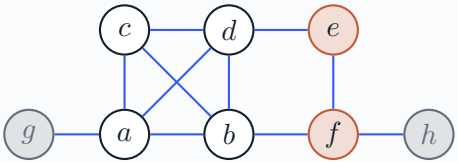
[12] Batagelj and Zaveršnik. arXiv, 2003.

The static k-core: peel low-degree vertices

k-core: the maximal subgraph in which every vertex has at least k neighbors *inside* the subgraph.

Core number of v: the largest k such that v is in the k-core.

Peeling: repeatedly remove a vertex of minimum degree. Its degree at removal (never below earlier values) is its core number.



core 1: g, h
core 2: e, f

[12] Batagelj and Zaveršnik. arXiv, 2003.

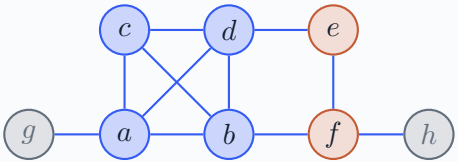
The static k-core: peel low-degree vertices

k-core: the maximal subgraph in which every vertex has at least k neighbors *inside* the subgraph.

Core number of v: the largest k such that v is in the k-core.

Peeling: repeatedly remove a vertex of minimum degree. Its degree at removal (never below earlier values) is its core number.

$O(n+m)$ with bucket queues.[12] Cores are nested: 3-core $\subseteq$ 2-core $\subseteq$ 1-core.

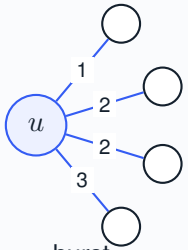


core 1: g, h
core 2: e, f
core 3: a, b, c, d

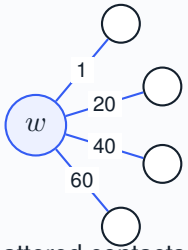
[12] Batagelj and Zaveršnik. arXiv, 2003.

Static cores ignore interaction timing

Degree counts neighbors in the aggregate graph, not when the interactions took place.



burst



scattered contacts

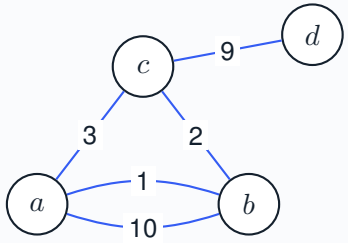
Both centers have degree 4. With $\Delta = 2$: every edge at u has 4 edges at u within ± 2; every edge at w has only itself.

A static core cannot distinguish a burst from scattered contacts.

(k, Δ)-cores count only nearby interactions

Δ-degree of a temporal edge $e = (\{u, v\}, t)$: the number of edges at u with time in $[t - \Delta, t + \Delta]$, minimized over both endpoints (counting e itself).

(k, Δ)-core: the maximal set of temporal edges in which every edge has Δ-degree at least k.[13]



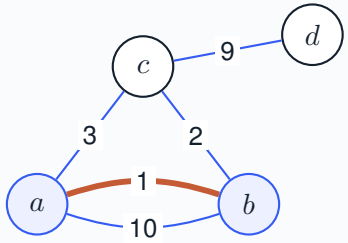
Edge labels: interaction times. $\Delta = 2$.

[13] **O.** et al. WSDM, 2025.

(k, Δ)-cores count only nearby interactions

Δ-degree of a temporal edge $e = (\{u, v\}, t)$: the number of edges at u with time in $[t - \Delta, t + \Delta]$, minimized over both endpoints (counting e itself).

(k, Δ)-core: the maximal set of temporal edges in which every edge has Δ-degree at least k.[13]



Edge labels: interaction times. $\Delta = 2$.

a–b at time 1: window $[-1, 3]$.

At a: times 1, 3 $\rightarrow$ 2 edges.
At b: times 1, 2 $\rightarrow$ 2 edges.

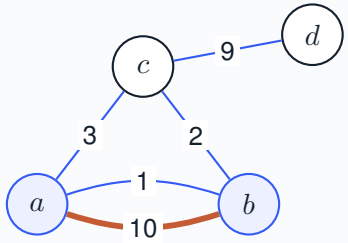
Δ-degree $= \min(2, 2) = 2$.

[13] **O.** et al. WSDM, 2025.

(k, Δ)-cores count only nearby interactions

Δ-degree of a temporal edge $e = (\{u, v\}, t)$: the number of edges at u with time in $[t - \Delta, t + \Delta]$, minimized over both endpoints (counting e itself).

(k, Δ)-core: the maximal set of temporal edges in which every edge has Δ-degree at least k.[13]



Edge labels: interaction times. $\Delta = 2$.

a–b at time 10: window $[8, 12]$.

At a and at b: only the edge itself.

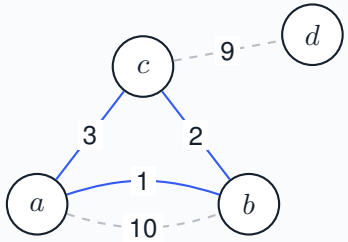
Δ-degree $= 1$, although a and b also met at time 1.

[13] **O.** et al. WSDM, 2025.

(k, Δ)-cores count only nearby interactions

Δ-degree of a temporal edge $e = (\{u, v\}, t)$: the number of edges at u with time in $[t - \Delta, t + \Delta]$, minimized over both endpoints (counting e itself).

(k, Δ)-core: the maximal set of temporal edges in which every edge has Δ-degree at least k.[13]



Edge labels: interaction times. $\Delta = 2$.

Peel with $k = 2$: the edges at times 9 and 10 drop out.

The $(2, 2)$-core is the triangle at times 1–3. The pair a–b is in the core at time 1, not at time 10.

For Δ at least the largest time span at a vertex, this is the static (edge) k-core.

[13] **O.** et al. WSDM, 2025.

Edge peeling computes all (k, Δ)-cores

1. Store each vertex's edges in time order; get Δ-degrees by binary search.
2. Bin-sort the temporal edges by Δ-degree.
3. Remove an edge of minimum Δ-degree; that value is its core number.
4. Decrement the Δ-degrees of its Δ-incident edges and update their bins.

Cost: $O(m \cdot \max(\log \delta_{\max}, \xi))$ time and $O(m)$ space, where ξ is the largest number of Δ-incident edges of an edge.[13]

Span-cores enumerate time intervals: $O(|\mathcal{T}|^2 \cdot m)$.[14]

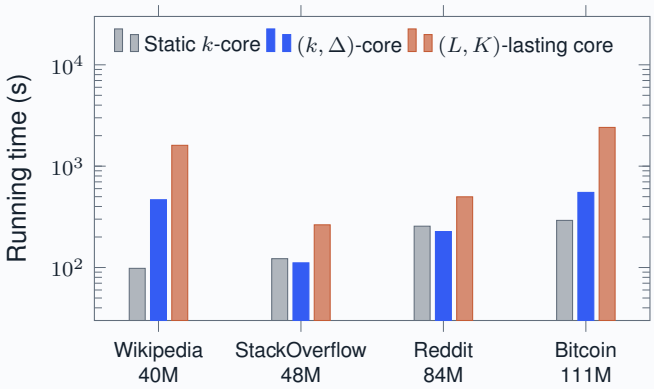
[13] **O.** et al. WSDM, 2025. [14] Galimberti et al. CIKM, 2018.

(k, Δ)-cores scale to 100M+ temporal edges

Eight real networks; the four largest are shown. Δ: median inter-event time.[13]

(k, Δ)-cores: 0.9–$4.8\times$ the static k-core time.

Baselines: (L, K)-lasting cores are 2–$4\times$ slower;[15] span-cores[14] run out of memory (96 GB) on all eight.

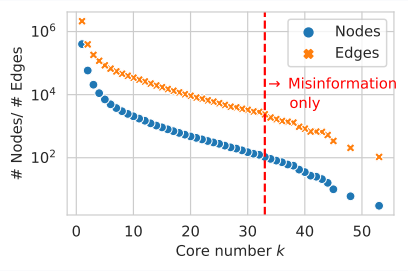


Temporal edges below each name.

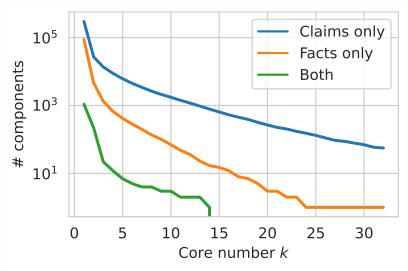
[13] **O.** et al. WSDM, 2025. [15] Hung and Tseng. DASFAA, 2021. [14] Galimberti et al. CIKM, 2018.

Inner cores hold only misinformation

Retweets in the six months before the 2016 US election: 346K users, 2.1M retweets, timestamps in seconds; 81.9% of retweets labeled misinformation, the rest fact-checking. $\Delta = 6.4$ minutes.[13]

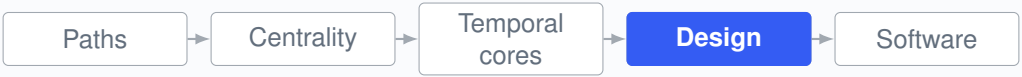


For $k \geq 33$, cores contain only misinformation.



For $k \geq 14$, no time-connected group mixes the two.

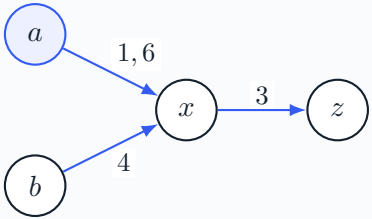
[13] **O.** et al. WSDM, 2025.



How should we change the schedule?

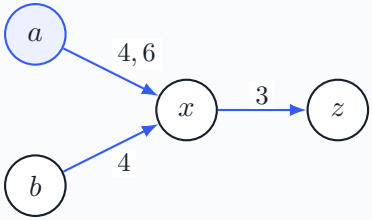
The timestamp swap was a design operation

Original record



a reaches z at time 4.

Delay one event: $a \to x$ from 1 to 4



a misses the departure $x \to z$ at 3.

A planner may *improve access*, *limit spread*, or *allocate* limited activations fairly.

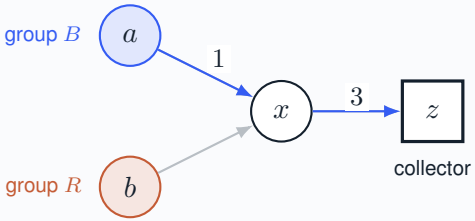
The aggregate graph stays the same. Temporal access changes.

Design problems differ in the allowed change

Allowed change	Question	Examples
Delay selected events	How does timing control reachability sets?	[16]
Shift temporal paths	How should connected schedules move together?	[17]
Choose activations	Can a small schedule meet coverage for every group?	**Fair Minimum Labeling**[18]

[16] Deligkas and Potapov. Information and Computation, 2022. [17] Deligkas et al. ESA, 2026.
[18] **O.** and Michail. NeurIPS, 2025.

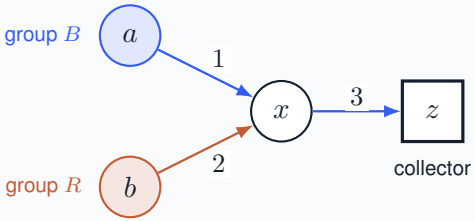
Choosing activations: the cheapest schedule serves one group



Now we **choose** the event times. Each activation costs 1.
Sources should reach the collector z.

Cheapest: 2 activations.
Only a reaches z; group R is left out.

Choosing activations: the cheapest schedule serves one group



Now we **choose** the event times. Each activation costs 1.
Sources should reach the collector z.

Cheapest: 2 activations.
Only a reaches z; group R is left out.

One source per group: 3 activations.
Now b reaches z as well.

Minimizing cost alone can exclude a whole group.

Fair Minimum Labeling asks for group-wise access at minimum cost

Input: a graph whose nodes belong to groups V_c, terminals, and a requirement q_c per group.

Find: a labeling $\lambda : E \to 2^{\mathbb{N}}$ with the fewest labels $|\lambda| = \sum_e |\lambda(e)|$ such that, in every group, at least q_c nodes reach the terminals by time-respecting paths.[18]

Hard

NP-complete even on a star with one terminal; no $(1-\epsilon)\ln(\#\text{groups})$-approximation.

But approximable

One terminal: reduction to Covering Steiner, 4-approximation. In general: a bicriteria approximation.

[18] **O.** and Michail. NeurIPS, 2025.

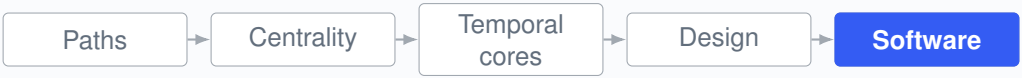
Fairness costs only four extra activations

Case study: random geometric graph, 1024 nodes, one learner as terminal; 64 sources in group B (near) and 64 in R (far), $q_B = q_R = 32$. The two groups carry different learning signals.[18]

		Coverage		Accuracy	
Method	**Cost**	B	R	B	R
Minimum cost, no groups	**64.0**	64.0	**0.0**	.996	**.408**
Closest source first	118.2	32.6	32.0	.943	.948
Bicriteria approximation	76.5	34.1	32.7	.950	.949
Covering Steiner (4-approx.)	68.0	32.0	32.0	.950	.948

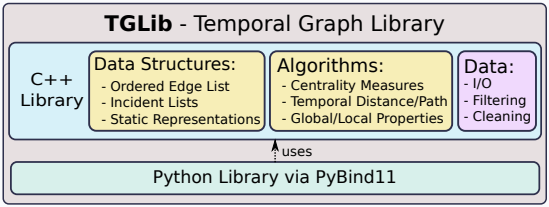
Four more activations (64 $\rightarrow$ 68) raise accuracy on group R from .41 to .95.

[18] **O.** and Michail. NeurIPS, 2025.



Where can you try this?

TGLib: a temporal graph library



Data structures: edge stream, edge incidence lists, time-respecting static graph, directed line graph, aggregate graph.

Algorithms: earliest arrival, fastest, shortest, minimum hops; closeness and top-k closeness, walk centrality, Katz, PageRank; burstiness, clustering, efficiency.

C++ header-only templates, OpenMP, Python via pybind11, MIT license.[19]

```
import pytglib as tgl
tgs = tgl.load_ordered_edge_list("data")
c = tgl.temporal_closeness(tgs,
      tgl.Distance_Type.Fastest)
```

`https://gitlab.com/tgpublic/tglib`

[19] **O.** and Mutzel. ICDM OPF, 2022.

Time order changes the answers

Takeaways

- **Paths:** event order decides what can reach what.
- **Centrality:** pruning and streaming make temporal scores fast.
- **Cores:** peeling temporal edges finds groups that interact closely in time.
- **Design:** moving one time can change access; fairness can be cheap.

Open directions

- Parallelism beyond many independent sources.
- Shared time semantics across tools: λ, strict vs. non-strict, waiting.
- Scalable methods for hard variants: restless paths, design.

Questions?

`https://lutzoe.github.io/` • `https://gitlab.com/tgpublic/tglib`

References (1/4)

[1] L. Oettershagen and P. Mutzel (2023). *An Index For Temporal Closeness Computation in Evolving Graphs*. SDM 2023. (preprint title: *An Index for Single Source All Destinations Distance Queries in Temporal Graphs*).

[2] F. Béres, R. Pálovics, A. Oláh, and A. A. Benczúr (2018). *Temporal walk based centrality metric for graph streams.* Applied Network Science 3, 32.

[3] B.-M. Bui-Xuan, A. Ferreira, and A. Jarry (2003). *Computing Shortest, Fastest, and Foremost Journeys in Dynamic Networks*. IJFCS 14(2), 267–285.

[4] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu (2014). *Path Problems in Temporal Graphs*. Proceedings of the VLDB Endowment 7(9), 721–732.

[5] A. Casteigts, A.-S. Himmel, H. Molter, and P. Zschoche (2021). *Finding Temporal Paths Under Waiting Time Constraints*. Algorithmica 83, 2754–2802.

References (2/4)

[6] M. Bentert, A.-S. Himmel, A. Nichterlein, and R. Niedermeier (2020). *Efficient computation of optimal temporal walks under waiting-time constraints*. Applied Network Science 5, 73.

[7] F. Brunelli and L. Viennot (2023). *Computing Temporal Reachability Under Waiting-Time Constraints in Linear Time*. SAND 2023, LIPIcs 257, 4:1–4:11.

[8] R. K. Pan and J. Saramäki (2011). *Path lengths, correlations, and centrality in temporal networks.* Physical Review E 84, 016105.

[9] L. Oettershagen and P. Mutzel (2022). *Computing top-k temporal closeness in temporal networks.* Knowledge and Information Systems 64, 507–535.

[10] L. Oettershagen, P. Mutzel, and N. M. Kriege (2022). *Temporal Walk Centrality: Ranking Nodes in Evolving Networks*. WWW 2022.

References (3/4)

[11] L. Oettershagen, N. M. Kriege, and P. Mutzel (2023). *A Higher-Order Temporal H-Index for Evolving Networks*. KDD 2023.

[12] V. Batagelj and M. Zaveršnik (2003). *An $O(m)$ Algorithm for Cores Decomposition of Networks.* arXiv:cs/0310049.

[13] L. Oettershagen, A. L. Konstantinidis, and G. F. Italiano (2025). *An Edge-Based Decomposition Framework for Temporal Networks*. WSDM 2025.

[14] E. Galimberti, A. Barrat, F. Bonchi, C. Cattuto, and F. Gullo (2018). *Mining (maximal) Span-cores from Temporal Networks.* CIKM 2018. Extended treatment with M. Ciaperoni: arXiv:1910.03645.

[15] W.-C. Hung and C.-Y. Tseng (2021). *Maximum (L, K)-Lasting Cores in Temporal Social Networks*. DASFAA 2021 International Workshops, LNCS 12680, 336–352.

References (4/4)

[16] A. Deligkas and I. Potapov. *Optimizing Reachability Sets in Temporal Graphs by Delaying.* AAAI 2020. Journal version: Information and Computation 285, 104890 (2022).

[17] A. Deligkas, M. Döring, E. Eiben, G. Skretas, and G. Tennigkeit (2026). *Maximizing Reachability via Shifting of Temporal Paths.* ESA 2026, LIPIcs 388, 122:1–122:16.

[18] L. Oettershagen and O. Michail (2025/2026). *Fair Minimum Labeling: Efficient Temporal Network Activations for Reachability and Equity*. NeurIPS 2025. Corrected and revised manuscript, September 2026.

[19] L. Oettershagen and P. Mutzel (2022). *TGLib: An Open-Source Library for Temporal Graph Analysis*. ICDM Open Project Forum 2022.

Photo credits and reuse terms

Photograph / source	Photographer and license
(20240630) Berlin Hbf 01.jpg	Roy Zuo. CC BY-SA 4.0.
Woman texting on a smartphone.jpg	Kristin Hardwick. CC0 1.0.
Community Networking Session.jpg	Pugal202. CC BY 4.0.
Cern datacenter.jpg	Hugovanmeijeren. CC BY-SA 3.0.

Sources: Wikimedia Commons. All four photographs are resized and cropped for the application panels. Source titles and license names above are clickable. The photographs and cropped views retain their stated licenses.