

Drawing Dynamic Graphs Without Timeslices

Paolo Simonetto¹, **Daniel Archambault**¹, and Stephen Kobourov²

¹Swansea University

²University of Arizona

September 27th, 2017



Swansea University
Prifysgol Abertawe

Computational Foundry
Ffowndri Gyfrifiadol

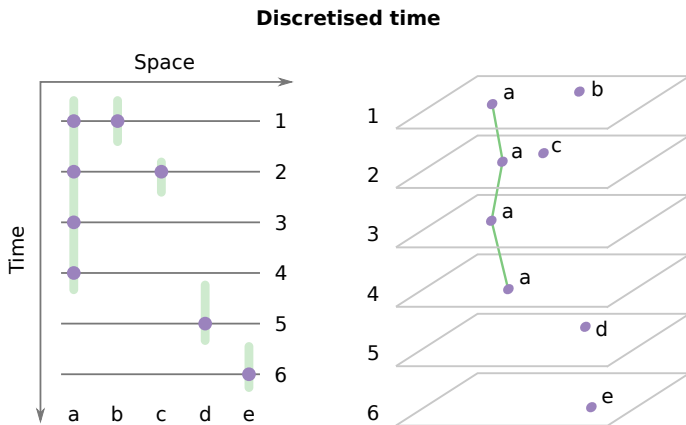
Outline

- 1 Introduction and Motivation
- 2 Related Work
- 3 Continuous Dynamic Graph Model
- 4 DynNoSlice Algorithm
- 5 Metric Evaluation
- 6 Conclusion and Future Work

Continuous Dynamic Graphs in the World

- Nodes and edges have real time values associated with them
 - ▶ streaming social media services (Twitter, Facebook, Weibo, ...)
 - ▶ social network data
 - ▶ experimental data
- Current methods transform them into discrete dynamic graphs by creating timeslices
- Dynamic graphs can also be naturally expressed using timeslices
 - ▶ standard timeslicing techniques work here

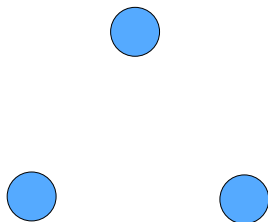
Discrete Dynamic Graph Drawing



- Timeslices selected or given in data
- Intertimeslice edge between same node in adjacent timeslices
- Linear interpolation between each timeslice
- Problem: How many timeslices to select?

Simple Temporal Pattern

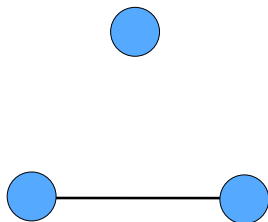
- Suppose in our data there is the following temporal pattern



- Timeslices are perfectly aligned with each event
 - ▶ In visualisation, we may not know a more complicated pattern exists
 - ▶ Computation of all possible patterns infeasible

Simple Temporal Pattern

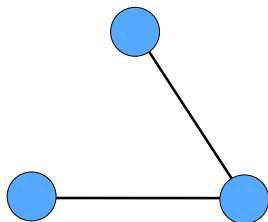
- Suppose in our data there is the following temporal pattern



- Timeslices are perfectly aligned with each event
 - ▶ In visualisation, we may not know a more complicated pattern exists
 - ▶ Computation of all possible patterns infeasible

Simple Temporal Pattern

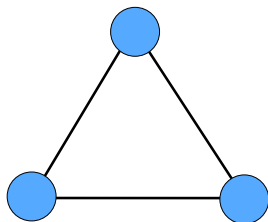
- Suppose in our data there is the following temporal pattern



- Timeslices are perfectly aligned with each event
 - ▶ In visualisation, we may not know a more complicated pattern exists
 - ▶ Computation of all possible patterns infeasible

Simple Temporal Pattern

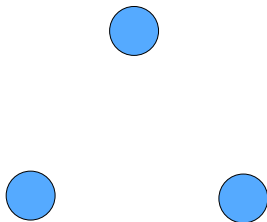
- Suppose in our data there is the following temporal pattern



- Timeslices are perfectly aligned with each event
 - ▶ In visualisation, we may not know a more complicated pattern exists
 - ▶ Computation of all possible patterns infeasible

Oversample Simple Temporal Pattern

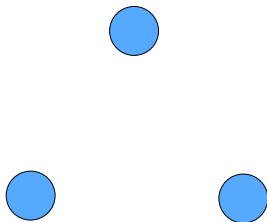
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



- It's like watching your data in extreme slow motion

Oversample Simple Temporal Pattern

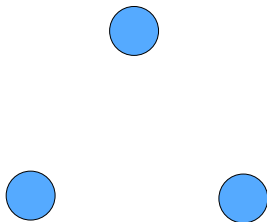
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



- It's like watching your data in extreme slow motion

Oversample Simple Temporal Pattern

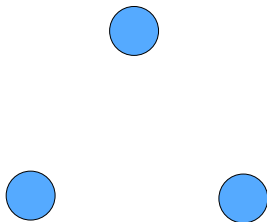
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



- It's like watching your data in extreme slow motion
 - ▶ Okay... any time now...

Oversample Simple Temporal Pattern

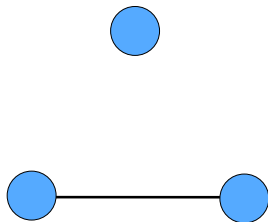
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



- It's like watching your data in extreme slow motion
 - ▶ Okay... any time now...

Oversample Simple Temporal Pattern

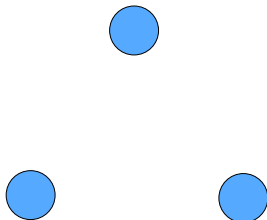
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



- It's like watching your data in extreme slow motion
 - ▶ Okay... any time now...
 - ▶ Yay! Now wait for the second edge...

Undersample Simple Temporal Pattern

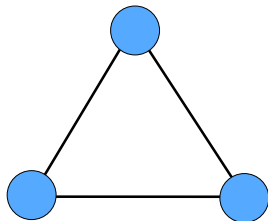
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

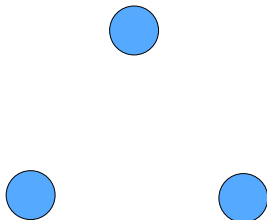
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

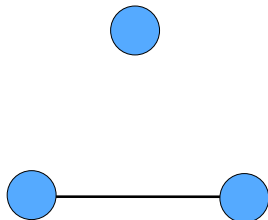
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

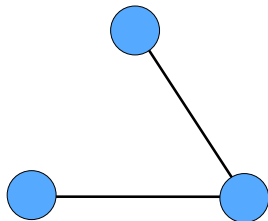
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

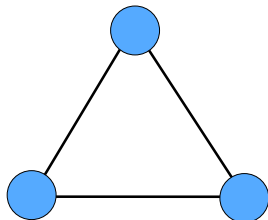
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

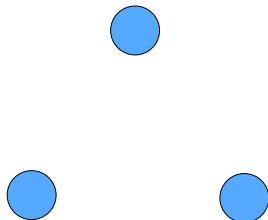
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

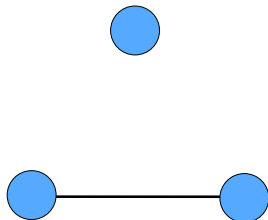
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

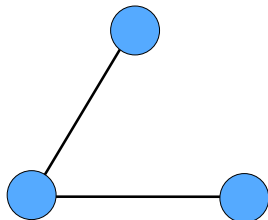
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

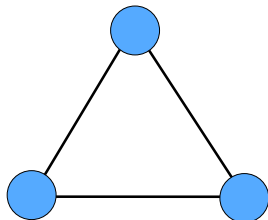
- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

Undersample Simple Temporal Pattern

- If we undersample, we lose temporal features



- Features are lost as we aggregate the time dimension
 - ▶ we cannot tell the difference between the two

The Problem Gets Worse...

- Both low and high frequency features can exist in a data set
 - ▶ Not much happens in the graph for several hours
 - ▶ Drastic changes over the course of 5 minutes
 - ▶ There is no single, regular timeslicing for this data
- Imposing regular timeslices forces instability in drawing
 - ▶ linear interpolations forced between adjacent timeslices
 - ▶ non-interacting nodes forced to have extra linear transitions
- Selecting a new set of timeslices means redrawing the network
 - ▶ one continuous drawing can be timesliced at any rate

Drawing Graphs Without Timeslices

- We propose a method to draw continuous dynamic graphs without timeslices:
 - ▶ A model for continuous dynamic graphs and their attributes
 - ▶ An algorithm that implements this model and draws continuous dynamic graphs in the space-time cube
 - ▶ A metric evaluation to demonstrate benefits of the approach

Related Work: Dynamic Graph Drawing







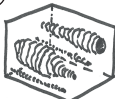

- Dynamic graph drawing approaches a mature area (Beck *et al.* 2017)
- Many approaches exist for online and offline drawing
 - ▶ supergraph, anchoring, and linking strategies are typical (Brandes and Mader 2012)
- Notion of timeslice often considered part of definition

A dynamic graph is defined as a sequence: $\Gamma = (G_1, G_2, \dots, G_n)$ where $G_i = (V_i, E_i)$ are static graphs and indices refer to a sequence of time steps $\tau = (t_1, t_2, \dots, t_n)$.

F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A Taxonomy and Survey of Dynamic Graph Visualization. *Computer Graphics Forum*, 36(1): 133–159, 2017.

- Our work does not assume a sequence of timeslices but a continuous time dimension

Related Work: Space-Time Cube

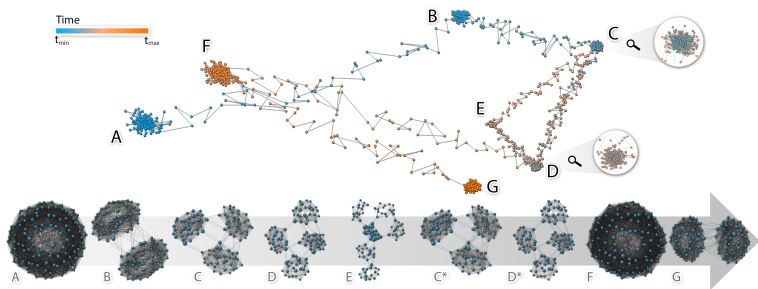
Density:	Dense		Sparse	
Variability:	Low	High	Low	High
Moving Objects	① 	② 	③ 	④ 
Matrix Data	⑤ 	⑥ 	⑦ 	⑧ 

B. Bach, P. Dragicevic, D. Archambault, C. Hurter and S. Carpendale. A Descriptive Framework for Temporal Data Visualizations Based on Generalized Space-Time Cubes. *Computer Graphics Forum*, 36(6):36–61, 2017.

- Related work uses the space-time cube to model visualisations
- Approaches do not describe how to draw continuous dynamic graphs in the space time cube

Related Work: Time as Space

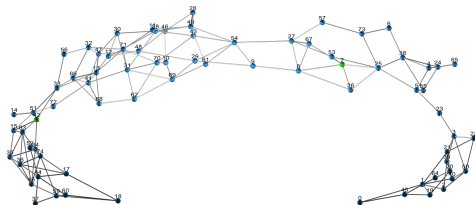
- Some approaches use one or two dimensions for time



S. van den Elzen, D. Holten, J. Blaas and J. J. van Wijk, Reducing Snapshots to Points: A Visual Analytics Approach to Dynamic Network Exploration, in *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1–10, 2016.

- Very scalable long time series
- Still, they are based on timeslicing

Related Work: 3D Graph Drawing

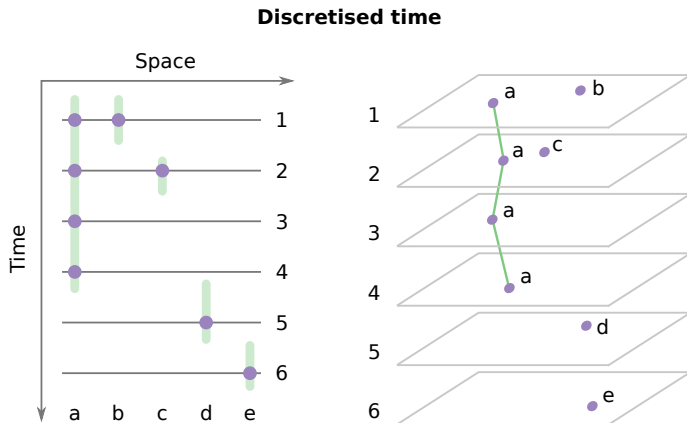


M. Cordeil, T. Dwyer, K. Klein, B. Laha, K. Marriott and B. H. Thomas, *Immersive Collaborative Analysis of Network Connectivity: CAVE-style or Head-Mounted Display?*, in *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 441-450, 2017.

- A number of algorithms draw graphs in 3D
 - ▶ have more “space” to draw the network
 - ▶ problems with occlusion and interaction
- Our approach 2D + time but is inspired by 3D approaches

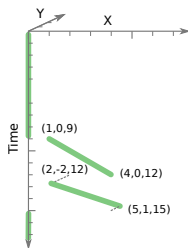
- 1 Model to formally describe continuous dynamic graphs
 - ▶ nodes and edges
 - ▶ attributes and how encoded
- 2 Algorithm to draw in 3D (2D + t) using this model (DynNoSlice)
 - ▶ force system comprising 5 forces
 - ▶ constraints, 3 of them, to ensure valid drawing
 - ▶ trajectory complexity adjustment

Discrete Dynamic Graph Model

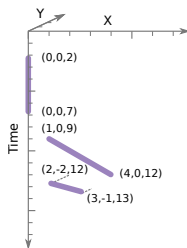


- Timeslices selected and nodes/edges projected onto nearest
- Static graph drawing on each timeslice
- Intertimeslice edges make stable drawings (mental map)

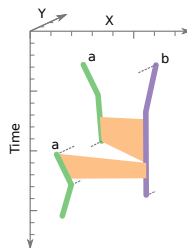
Continuous Dynamic Graph Model



(a)



(b)



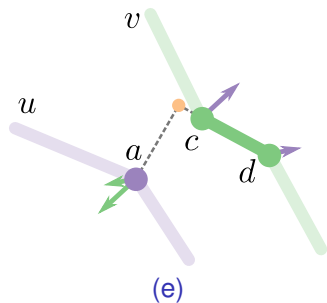
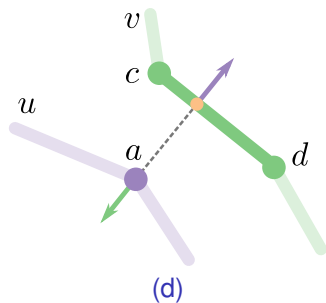
(c)

- A continuous dynamic graph in the 3D space-time cube:
 - ▶ nodes are polyline trajectories with bends
 - ▶ edges are ruled surfaces between two polyline trajectories
 - ▶ attributes are assigned to both over intervals
- Positions are 3D coordinates (x, y, t)
- Nodes, edge, attributes all defined over intervals of time

DynNoSlice Algorithm Overview

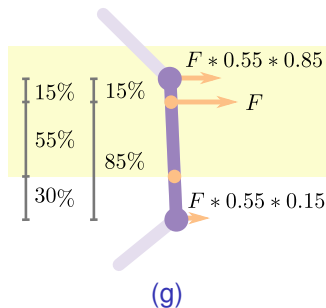
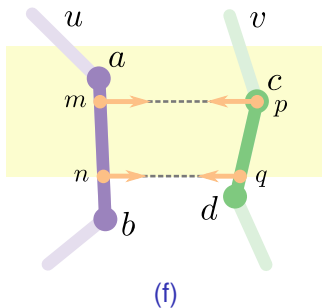
- Algorithm designed to embed polyline trajectories inside the space-time cube
- Output is 2D + time embedding of the nodes which are polylines
- For each iteration of the algorithm:
 - 1 Compute and sum the forces based on the force system.
 - 2 Move nodes based on these forces and the constraints.
 - 3 Adjust trajectory complexity in the space-time cube.

Node Repulsion



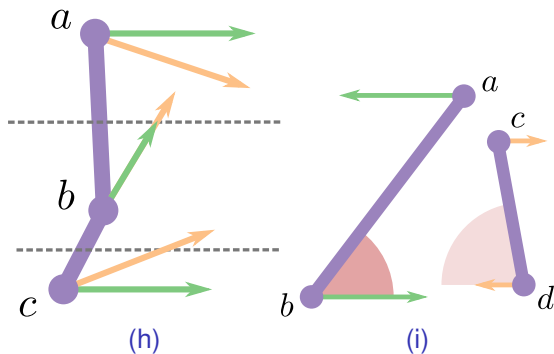
- Node trajectories u and v repel in 3D
 - ▶ fixed time points move in x and y
 - ▶ bends in trajectory can move up and down
- Spread trajectories in space and prevents crowding

Edge Attraction



- Surface pulls two trajectories u and v together
 - ▶ area edge occupies in time dimension
 - ▶ allocate force to node via linear interpolation

Gravity, Trajectory Straightening, and Stability



- Three forces to keep drawing properties
 - ▶ gravity pulls trajectories to centre of (x, y) drawing
 - ▶ create smooth trajectories using curve shortening flow
 - ▶ stable drawings (mental map) to encourage vertical segments
 - ★ force applied even when appearance is false

Constraints

- Constraints needed to ensure a valid drawing
 - 1 *Decreasing Max Movement.* large movements at start and small movements at end
 - 2 *Movement Acceleration.* oscillations discouraged between iterations
 - 3 *Time Correctness.* appearance/disappearance times must remain the same

Node Trajectory Complexity Adjustment

- Nodes are polylines in the space time cube
- We want to adjust trajectory complexity as needed
 - ▶ no interaction, remove bends to make straight
 - ▶ lots of interaction, insert bends for a more complicated trajectory
 - ▶ bends are inserted locally
 - ▶ inserted bends can change their time coordinate
- Adapt polyline complexity by inserting and removing bends
 - ▶ segment length is above a threshold, a bend is inserted
 - ▶ segment length is below a threshold, a bend is removed
- Inserting bend is like inserting a personalised timeslice for a particular node that needs it

- Metric evaluation comparing DynNoSlice to Visone
 - ▶ test both discrete and continuous data sets
 - ▶ on timeslice stress and off timeslice stress
 - ▶ node movement
 - ▶ crowding events
 - ▶ running time
- Further details in the paper on procedure and data sets

- Temporal resolution improves time features
- Network features made clear without sampling

Discrete Dynamic Graph Data Sets

Graph	Type	Time (s)	Scale	StressOn (d)	StressOff (d)	Movement	Crowding
VanDeBunt	v	0.13	1.00	1.14	1.46	3.80	0
	d	7.73	0.62	1.20	1.20	3.91	0
	c	6.73	0.68	1.19	1.29	3.69	0
Newcomb	v	0.11	1.00	14.04	14.77	16.36	8
	d	9.68	0.68	16.61	16.59	13.48	2
	c	7.62	0.75	18.13	17.98	12.37	0
InfoVis	v	77.43	0.47	51.66	52.98	2.15	36
	d	388.38	0.56	31.09	31.04	2.03	8
	c	381.15	0.56	32.70	34.02	1.91	6

- Visone is better for on timeslice stress
- Movement and crowding are comparable
- Some improvements off timeslice

Continuous Dynamic Graph Data Sets

Graph	Type	Time (s)	Scale	StressOn (c)	StressOff (c)	Movement	Crowding
Rugby	v	0.08	0.68	3.08	2.71	25.47	6
	d	7.40	0.68	1.84	1.71	16.23	1
	c	3.88	0.51	1.84	1.77	6.57	0
Pride	v	3.39	0.18	0.62	0.88	5.44	682
	d	1655.50	0.32	0.82	0.86	6.95	13
	c	75.61	0.24	0.83	0.85	1.12	3

- DynNoSlice has lower off-timeslice stress
- Lower movement and crowding
 - ▶ adaptive trajectory complexity allows all three

Conclusions and Future Work

- First dynamic graph drawing algorithms that does not use timeslices
 - ▶ Nodes modelled as polylines and edges as surfaces
 - ▶ Implemented the first algorithms to embed in space-time cube
- Comparison with timeslicing algorithms
- Animation is natural but not effective. New visualisation methods needed.
- Nyquist frequency and can we sample better?