Sampling for Effective Visual Network Analytics

Daniel Archambault[1]

[1]Swansea University

June 4th, 2018

Outline

- Impact of graph sampling on visualisation
  - machine learning/data mining will be applied
  - how does it influence visualisations
- Dynamic network visualisation without timeslices

Do Sampling Methods Influence Visualisation?

- Graph sampling methods developed in data mining/graph mining literature
- Reduce scale of the data and preserve statistics about graph
- People are going to use them to process network data
- What influence will it have on the visualisation?
  - Will high degree nodes still be perceived as high degree in the sample?
  - Will clusters still be perceived as clusters?
  - Is the coverage of the data still “good”?

Y. Wu, N. Cao, D. Archambault, Q. Shen, H. Qu and W. Cui, "Evaluation of Graph Sampling: A Visualization Perspective," in IEEE Transactions on Visualization and Computer Graphics (InfoVis 2017), vol. 23, no. 1, pp. 401-410, Jan. 2017.

Sampling: Common Way to Reduce Graph Size

- Many sampling methods exist in graph mining
- A subset of nodes and edges of the graph selected
  - usually this subset is representative of the graph
  - in graph mining, similar metrics (degree, clustering coef. ...)
- People are going to apply these algorithms and visualise samples
- What effects does this have on how the visualisation is perceived?

Sampling Methods Tested

- Methods drawn from (Leskovec and Faloutsos (KDD 2006))
  - Random Node (RN): random nodes + connecting edges
  - Edge Node (REN): random edges + nodes + connecting edges
  - Random Walk (RW): all nodes and edges on a random walk
  - Random Jump (RJ): RW + randomly jump to nodes on occasion
  - Forest Fire (FF): burn edges from seeds in geometric sequence
- Methods span many types of sampling algorithms
- Performed well in graph mining study

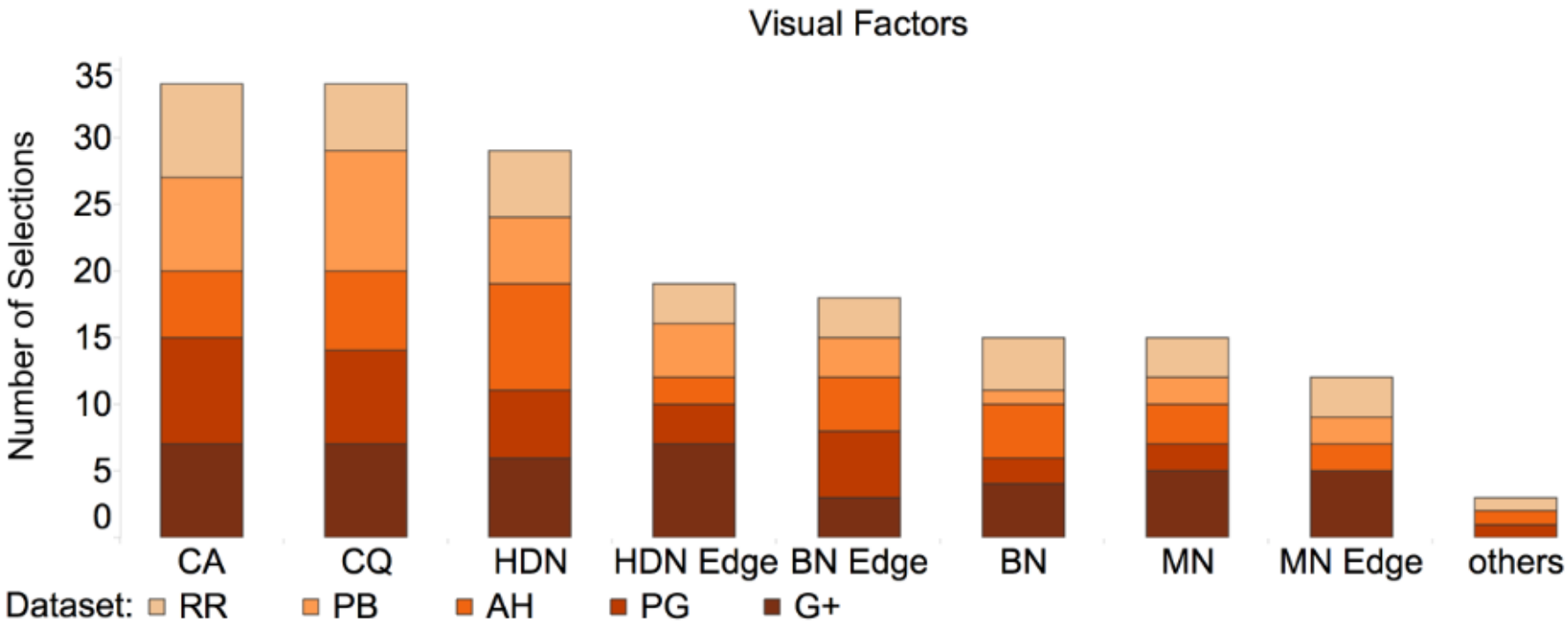
Experimental Procedure

- Run four experiments
- Part 1: pilot to determine which factors important
- Part 2: run three experiments to determine best performance
- Sampling methods applied to data sets
- Measure effect on node-link visualisations of those data sets

Pilot: What Visual Factors are Important?

- Show a number of sampling rates to determine lowest possible
- Ask participants what factors important (text box)
- All sampling algorithms tested on real networks

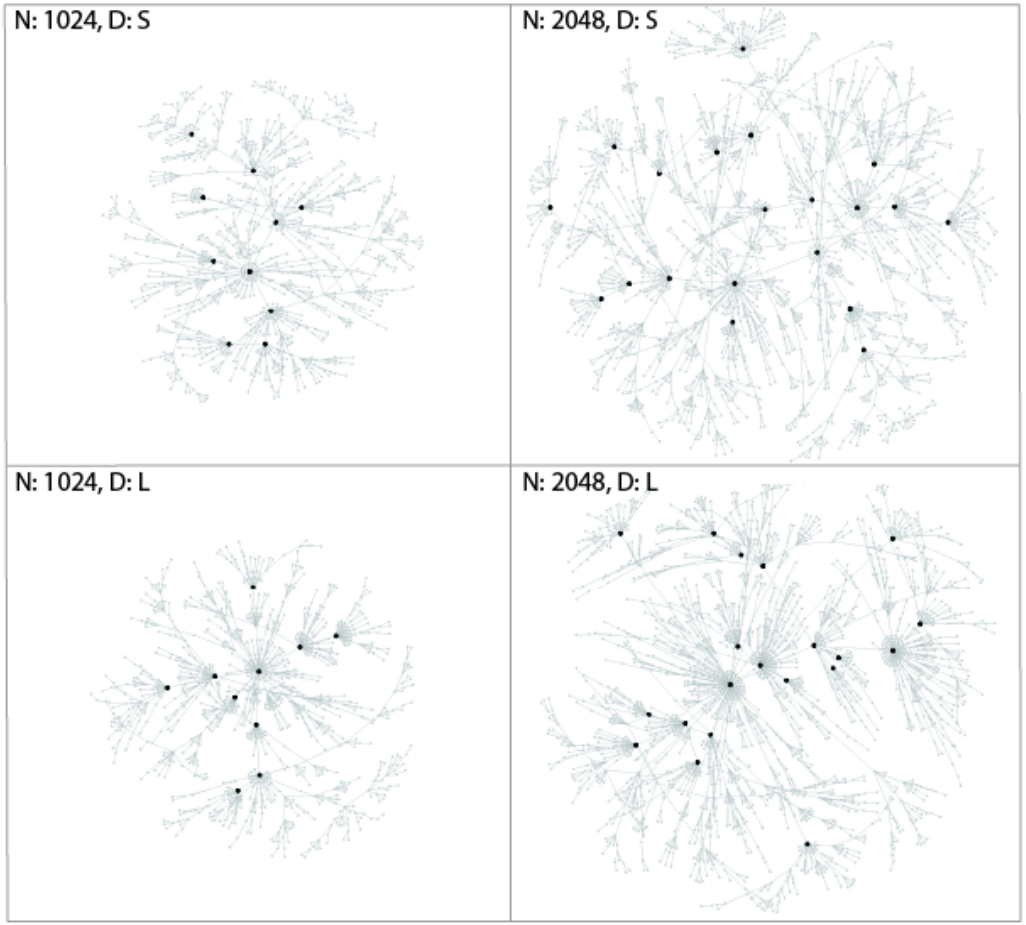
Results of Pilot



- Lowest sampling rate with good quality: 20% chosen
- Important visual factors for this judgement:
  - Coverage area - how “complete” the sample is
  - Cluster quality - are clusters well preserved
  - High degree nodes - are nodes still of high degree?

Experiment I: High Degree Nodes

- Do the high degree nodes still appear high degree in the sample?
- Participants select nodes that appear high degree in the sample
- How many of them are actually high degree?



Experiment I: Results

- Count of high degree nodes preserved in sample
  - REN keeps the most, RW the fewest
- Perception of high degree nodes
  - If RW selects it, it is generally perceived as high degree
  - Random walks can accentuate high degree in sample
  - RJ, REN, and FF are good but not as good as RW

Experiment II: Clusters

- Which method best preserves clusters in the sample?
- Unsampled in upper left
- Rate how each sampling strategy did (out of 5)

Experiment II: Results

- Perception of cluster quality in samples
  - REN and RJ perform best in perceived cluster quality
  - RJ - is similar to community finding (InfoMap)
  - Other sampling methods can miss clusters
  - Cluster number seems to be most important factor

Conclusions

- Sampling method influences perception of graph properties
- Results are different from metric measurements
- Important to consider how users perceive graph samples when applying graph mining methods to the data

Outline

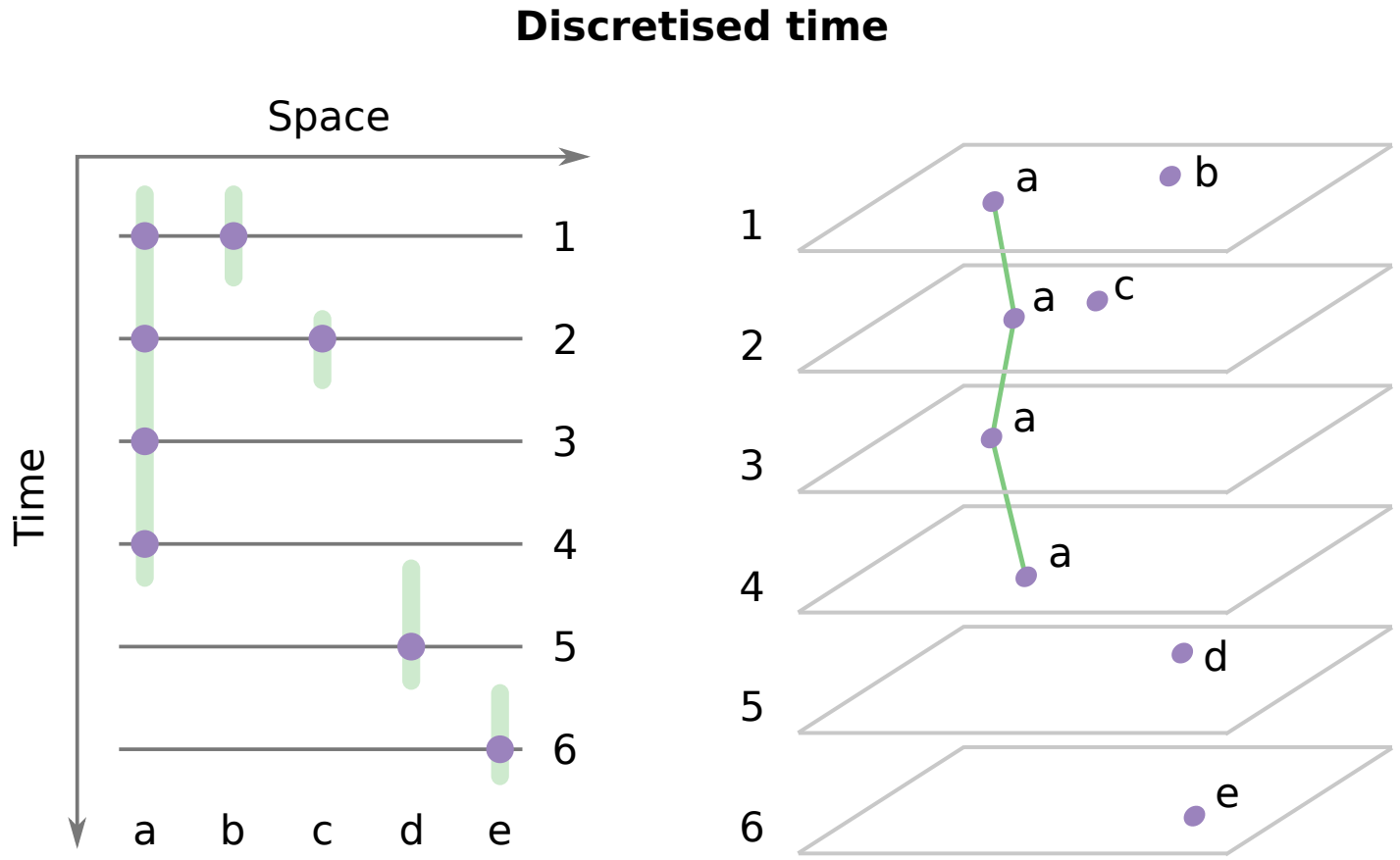
- Impact of graph sampling on visualisation
- Dynamic network visualisation without timeslices
  - you can draw event-based/streaming dynamic graphs offline
  - timeslicing dynamic graphs is a sampling problem

Event-Based 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
- We propose drawing the event-based data directly

Paolo Simonetto, Daniel Archambault, Stephen Kobourov. Proceedings of the 25th International Symposium on Graph Drawing and Network Visualization (GD 2017).

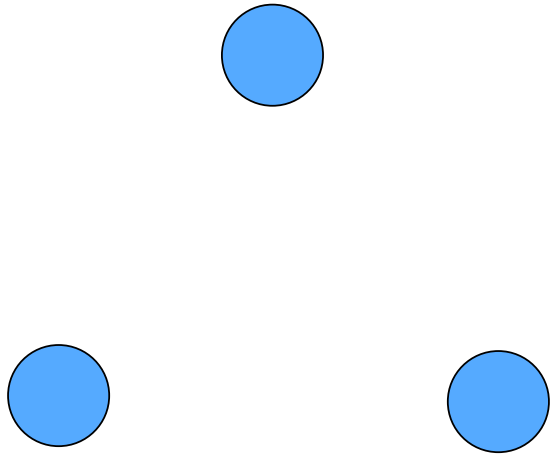
Timeslice-Based Graph Drawing



- 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

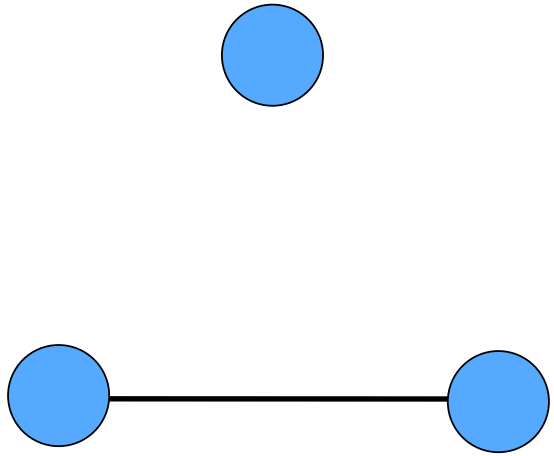
- Suppose in our data there is the following temporal pattern



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

Simple Temporal Pattern

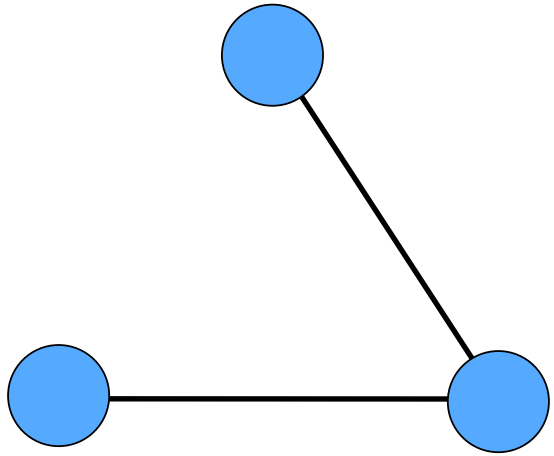
- Suppose in our data there is the following temporal pattern



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

Simple Temporal Pattern

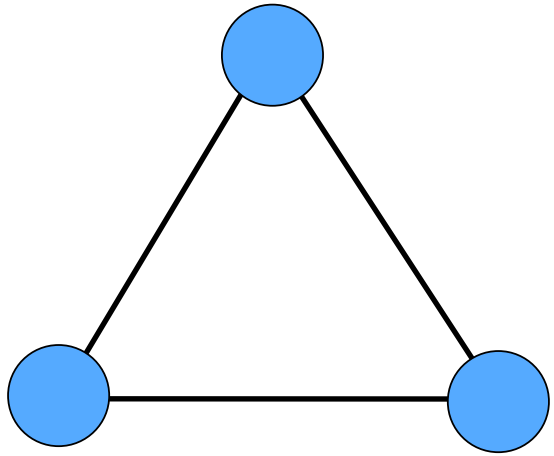
- Suppose in our data there is the following temporal pattern



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

Simple Temporal Pattern

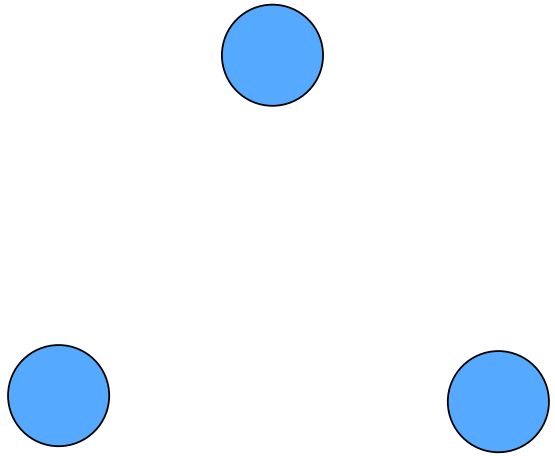
- Suppose in our data there is the following temporal pattern



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

Oversample Simple Temporal Pattern

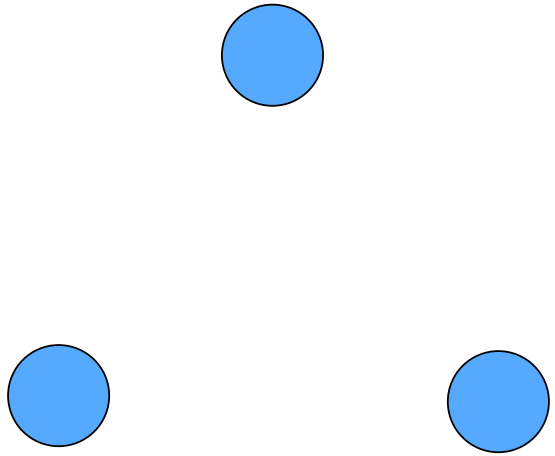
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



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

Oversample Simple Temporal Pattern

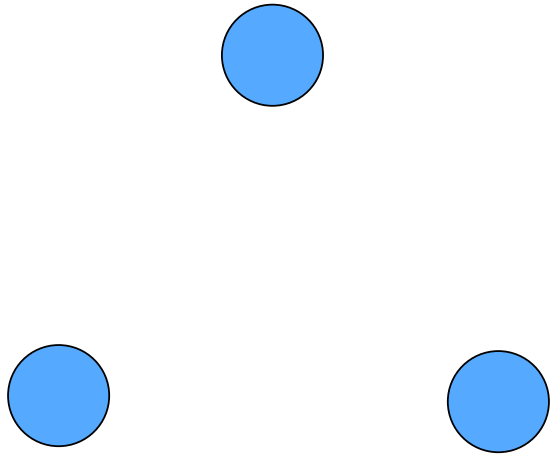
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



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

Oversample Simple Temporal Pattern

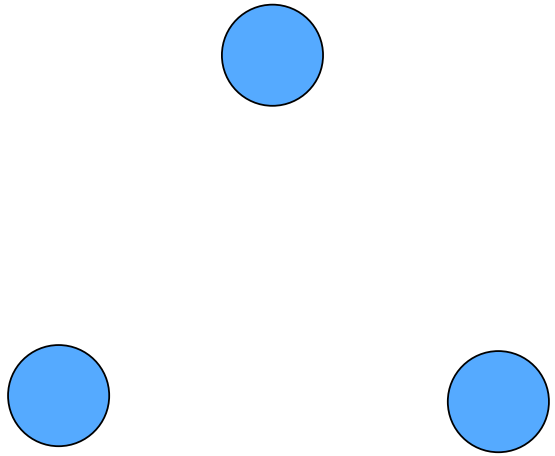
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



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

Oversample Simple Temporal Pattern

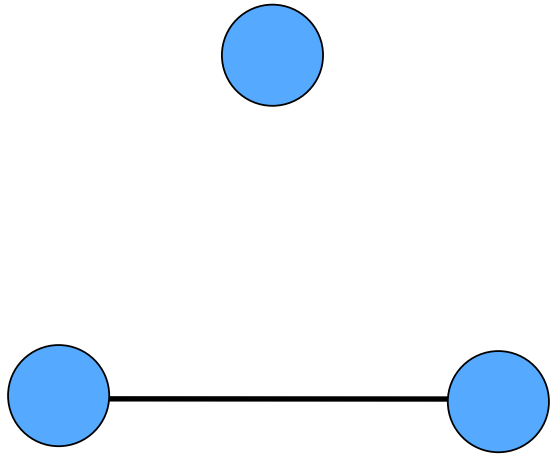
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



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

Oversample Simple Temporal Pattern

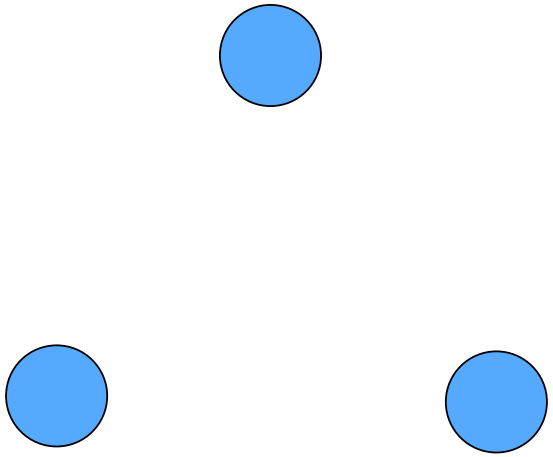
- If we oversample, we waste computational time
- We waste screenspace in small multiples and time in animation



- 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

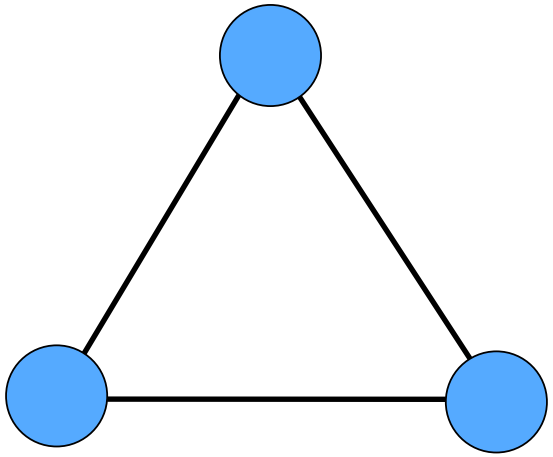
- If we undersample, we lose temporal features



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

Undersample Simple Temporal Pattern

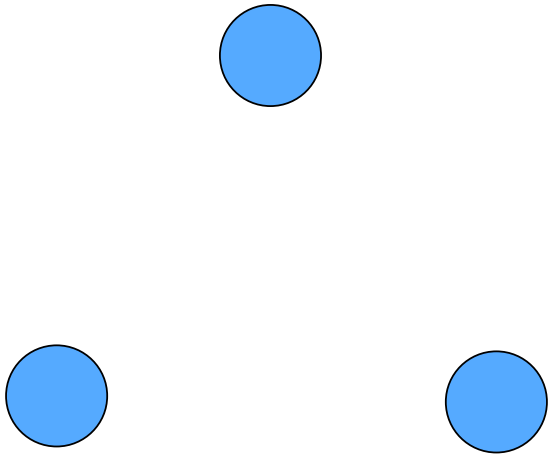
- If we undersample, we lose temporal features



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

Undersample Simple Temporal Pattern

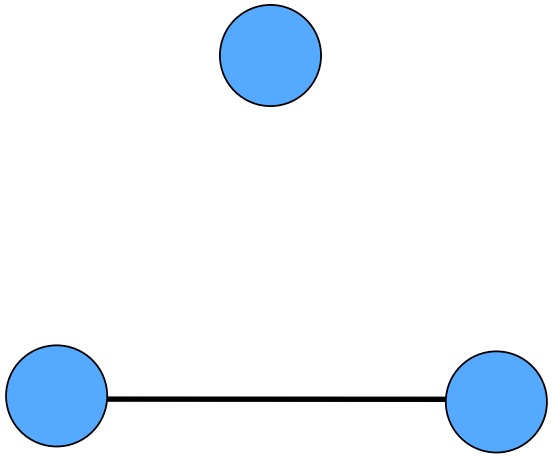
- If we undersample, we lose temporal features



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

Undersample Simple Temporal Pattern

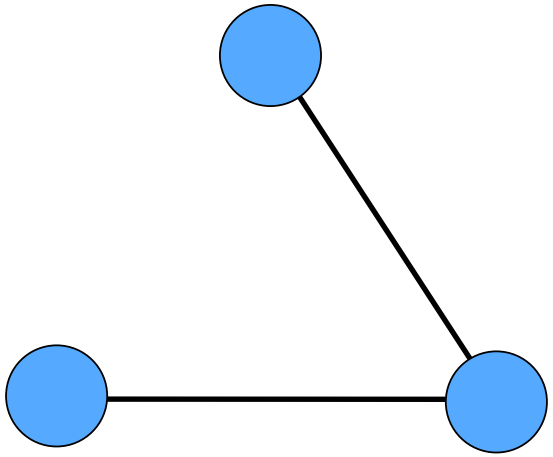
- If we undersample, we lose temporal features



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

Undersample Simple Temporal Pattern

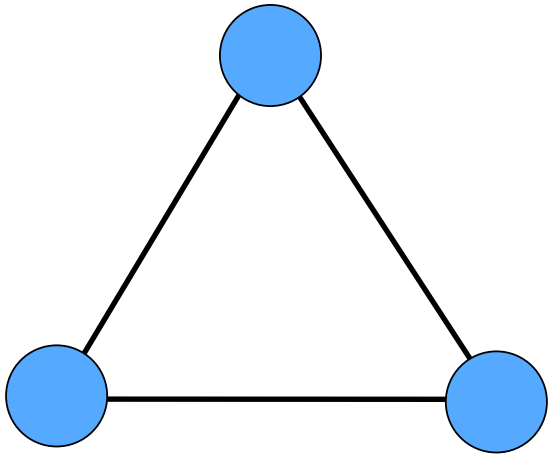
- If we undersample, we lose temporal features



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

Undersample Simple Temporal Pattern

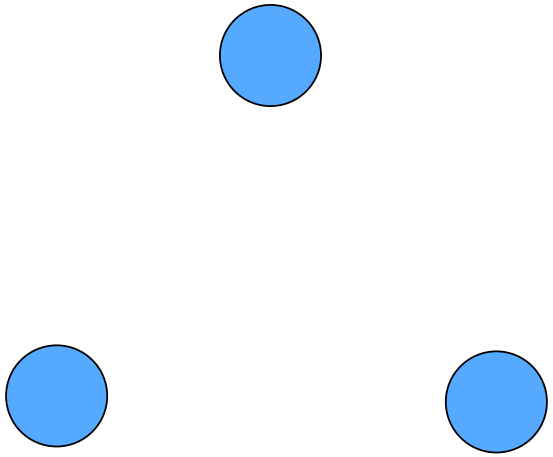
- If we undersample, we lose temporal features



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

Undersample Simple Temporal Pattern

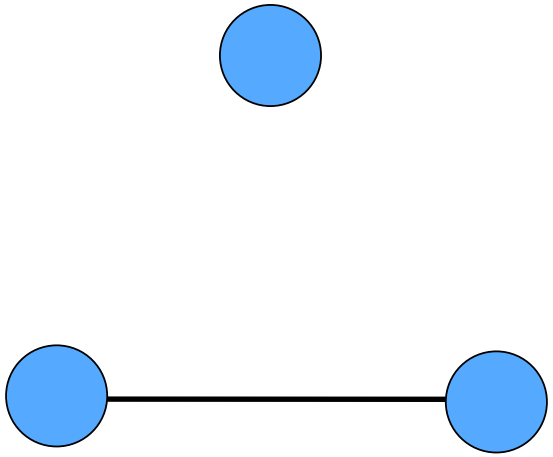
- If we undersample, we lose temporal features



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

Undersample Simple Temporal Pattern

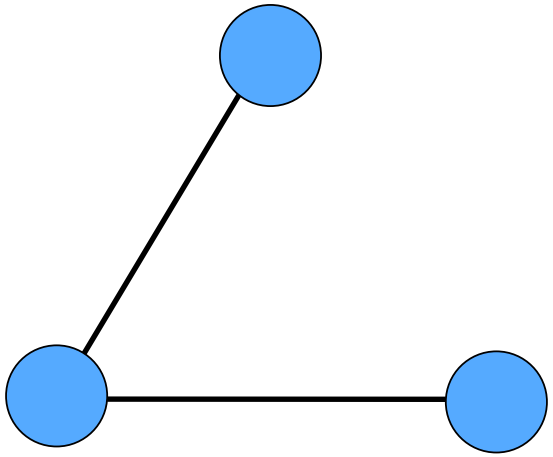
- If we undersample, we lose temporal features



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

Undersample Simple Temporal Pattern

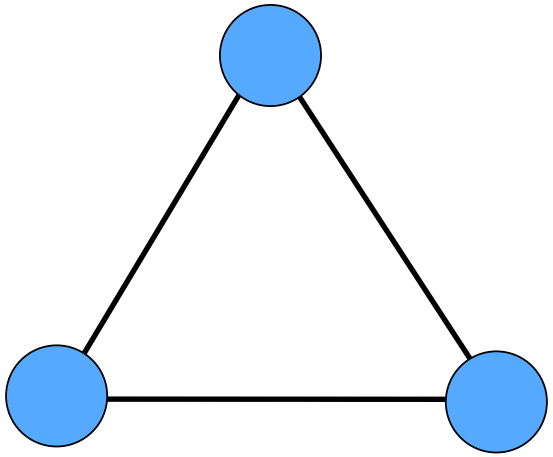
- If we undersample, we lose temporal features



- 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

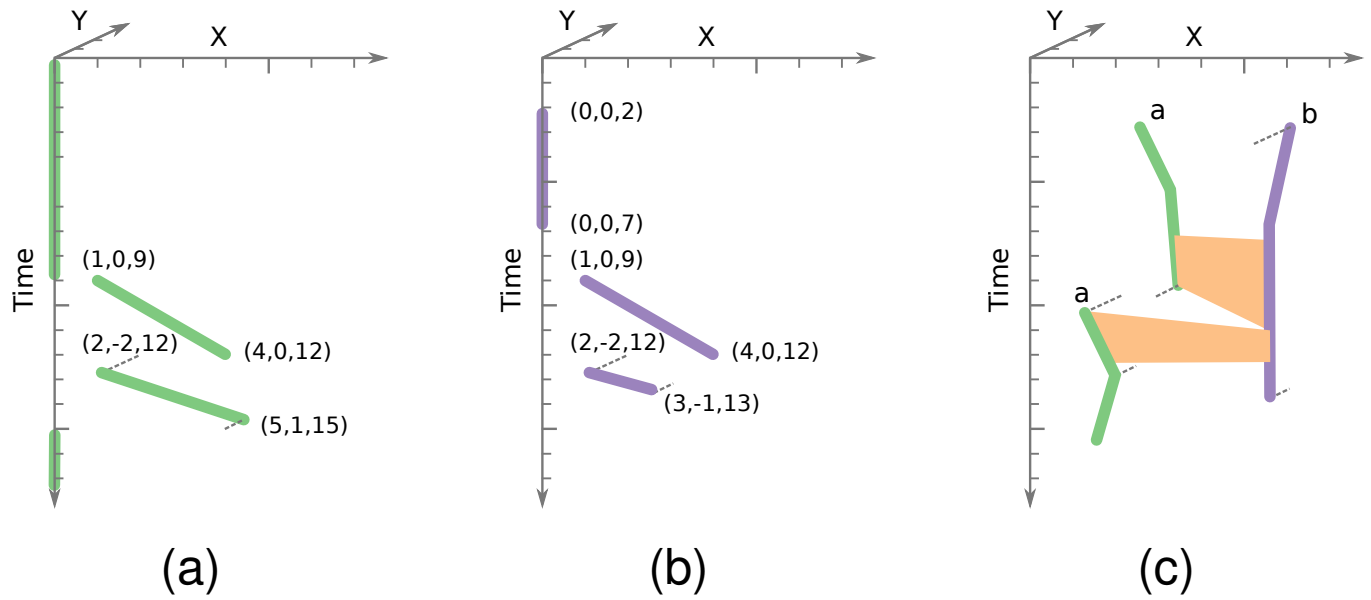
The Problem Gets Worse...

- 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 drawing can be timesliced at any rate

Event-Based Dynamic Graph Drawing

1. Model to formally describe event-based 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

Event-Based Dynamic Graph Model

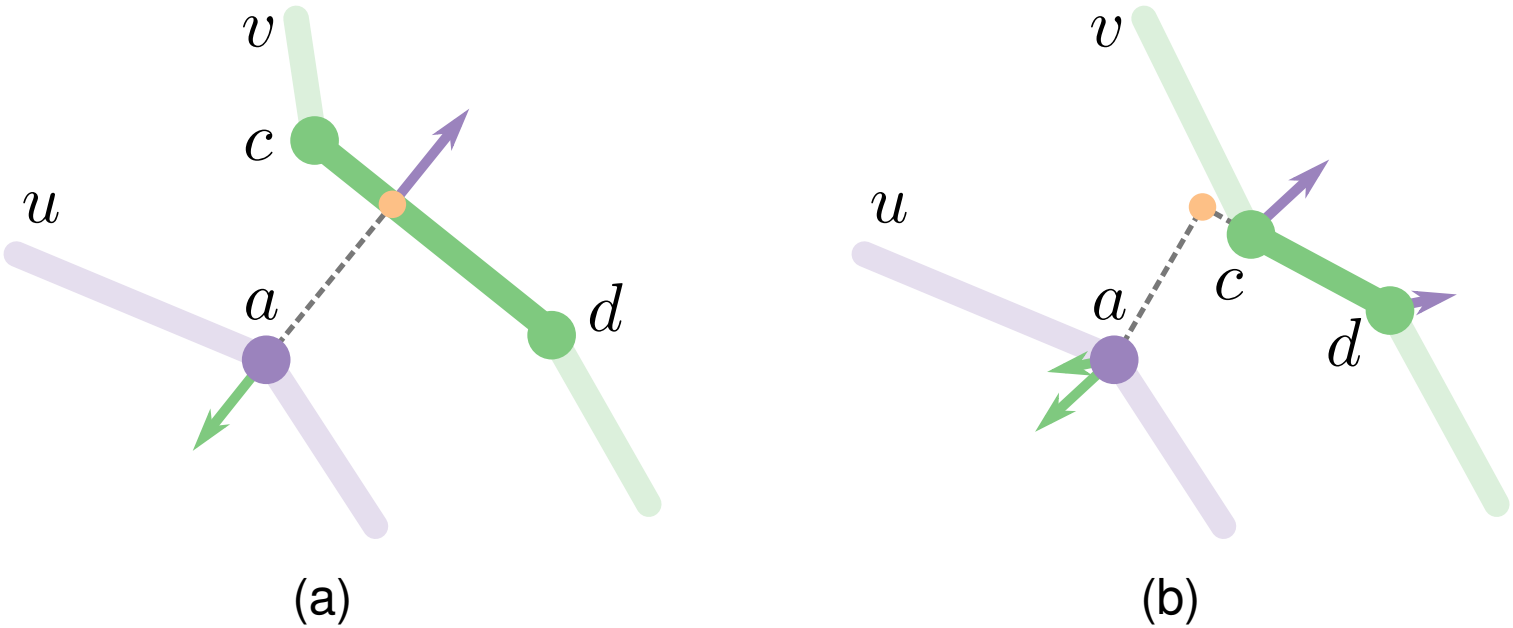


- An event-based dynamic graph in the 3D space-time cube:
  - nodes are polyline trajectories with bends
  - edges are ruled surfaces between two polyline trajectories
  - attributes are assigned to both over intervals
- Positions are 3D coordinates (x, y, t)
- Nodes, edge, attributes all defined over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

DynNoSlice Algorithm Overview

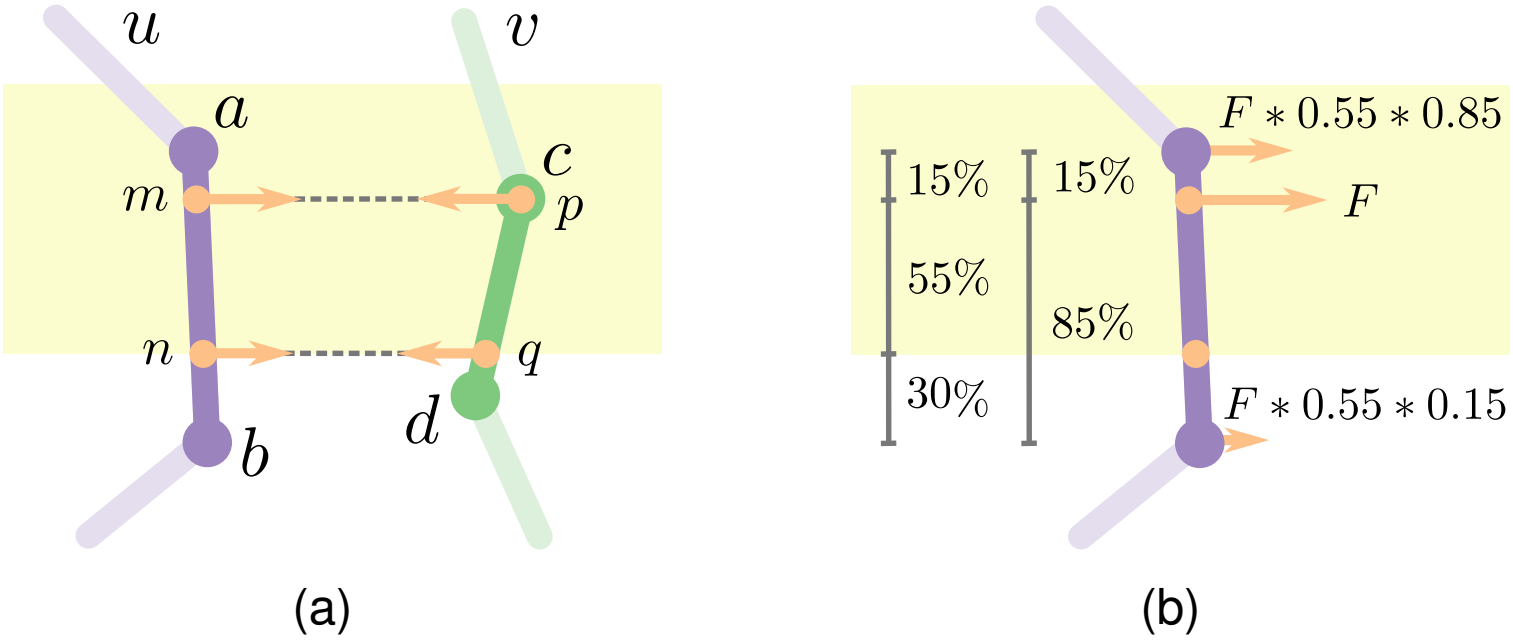
- Algorithm designed to embed polyline trajectories inside the space-time cube
- Output is 2D + time embedding of the nodes which are polylines
- For each iteration of the algorithm:
  1. Compute and sum the forces based on the force system.
  2. Move nodes based on these forces and the constraints.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

Evaluation

- Metric evaluation comparing DynNoSlice to Visone
  - node movement
  - crowding events
  - running time
- On event-based data, DynNoSlice outperforms timeslicing methods

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?

Summary

- Sampling influences how data is visualised
  - choose the right graph sampling method for your visualisation
  - sometimes it is better not to sample across time
- Sampling becomes commonplace with increased data size
- Critical to study its effects on visualisation