Dynamic Network Embeddings: From Random Walks to Temporal Random Walks

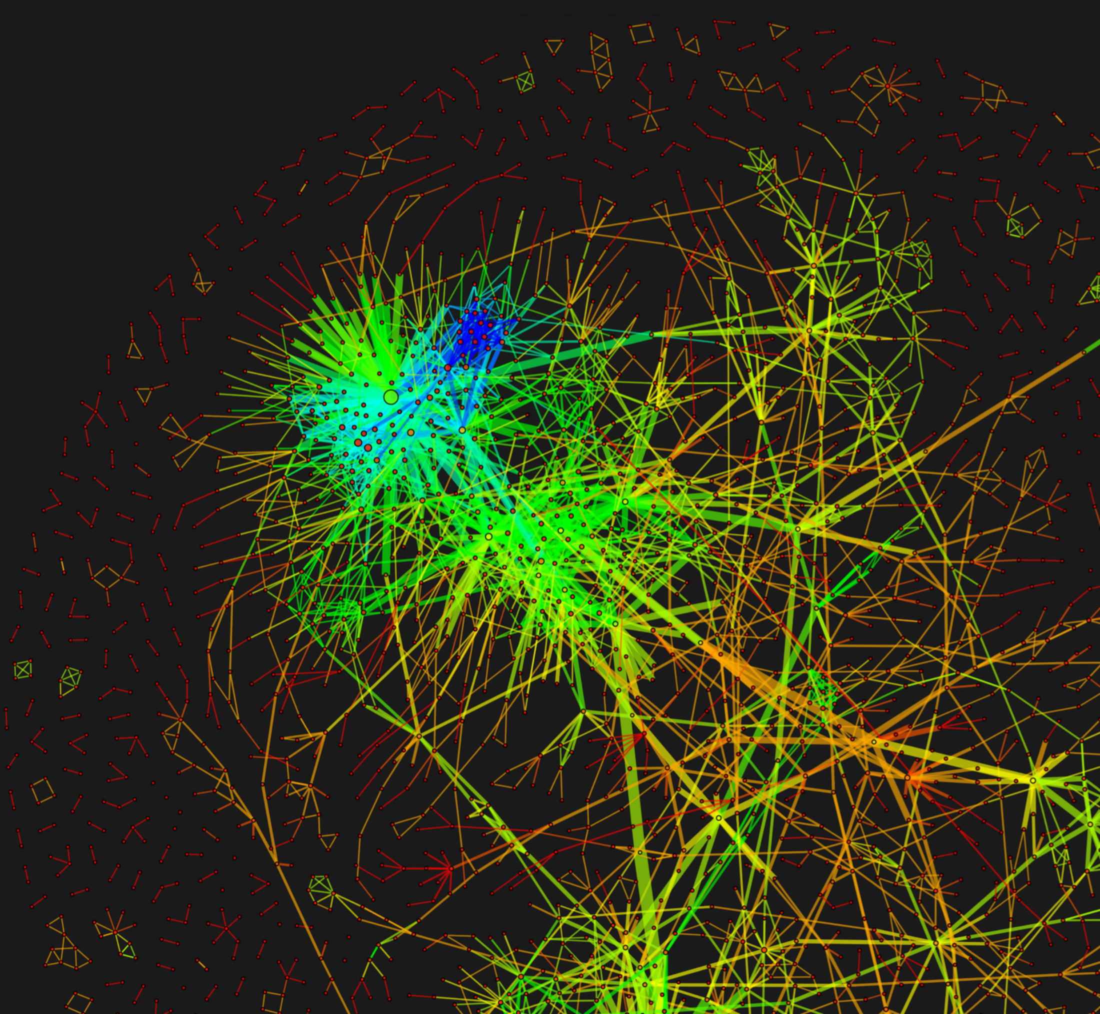
Ryan A. Rossi
Adobe Research

Joint work with:

Giang Hoang Nguyen
John Boaz Lee
WPI

Nesreen K. Ahmed
intel LABS

Eunyee Koh
Sungchul Kim
Adobe Research



Representation Learning in Graphs

- **Goal:** Learn representation (features) for a set of graph elements (nodes, edges, etc.)

$$\text{Given } G = (V, E)$$
$$\text{Learn a function } f : V \to \mathbb{R}^d$$

- **Key intuition**: Map the graph elements (e.g., nodes) to the d-dimension space, while preserving some type of "*similarity*", e.g., based on **proximity** (communities), or **structural similarity** (roles)
- Use the features for any downstream prediction task

Limitations of Current Methods

- Ignore temporal information (edge timestamps)
  - Most real-world networks are dynamic (evolve over time)

Some recent work uses **discrete static snapshot graphs** [Hisano, 2016; Kamra et al., 2017]

- Very coarse approximation & introduces noise/errors
- Temporally invalid
- Unclear how to create discrete snapshot graphs & differs for each network [Soundarajan et al., 2016]
- Time period to use depends highly on the underlying domain/application (NP-hard problem)

Problem:
Learn **Time-respecting** Embeddings from CTDN

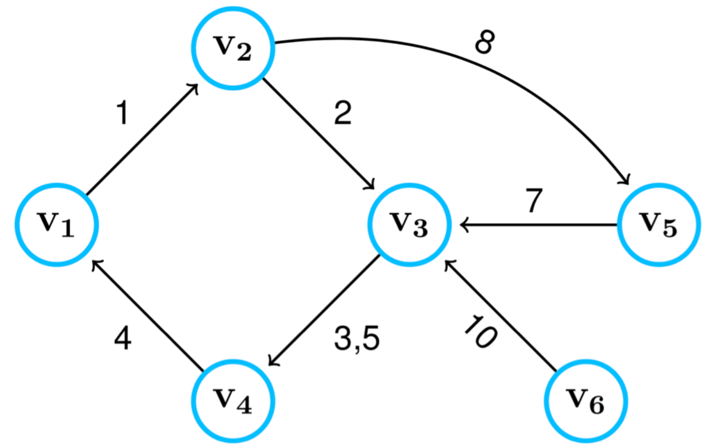
Goal: Find a mapping of nodes to a D-dimensional *time-dependent* representation

Properties warranted by approach:

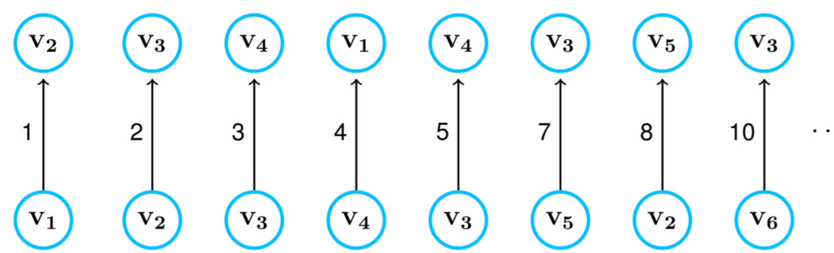
- Temporally valid
- Model network in the most natural way with min information loss
  - Continuous-time dynamic network (as opposed to a sequence of static snapshot graphs)
- General & unifying framework

Continuous-Time Dynamic Network Embeddings (CTDNEs)

- Temporally valid
- Model network at the finest temporal granularity
- Natural way to handle dynamic networks
  - Avoids noise/information loss with discrete static snapshot approaches
- Supports learning in graph streams where edges arrive continuously over time (e.g., every second/millisecond)



$$G = (V, E_T, \mathcal{T})$$
$$E_T \subseteq V \times V \times \mathbb{R}^+$$
$$\mathcal{T} : E \to \mathbb{R}^+$$

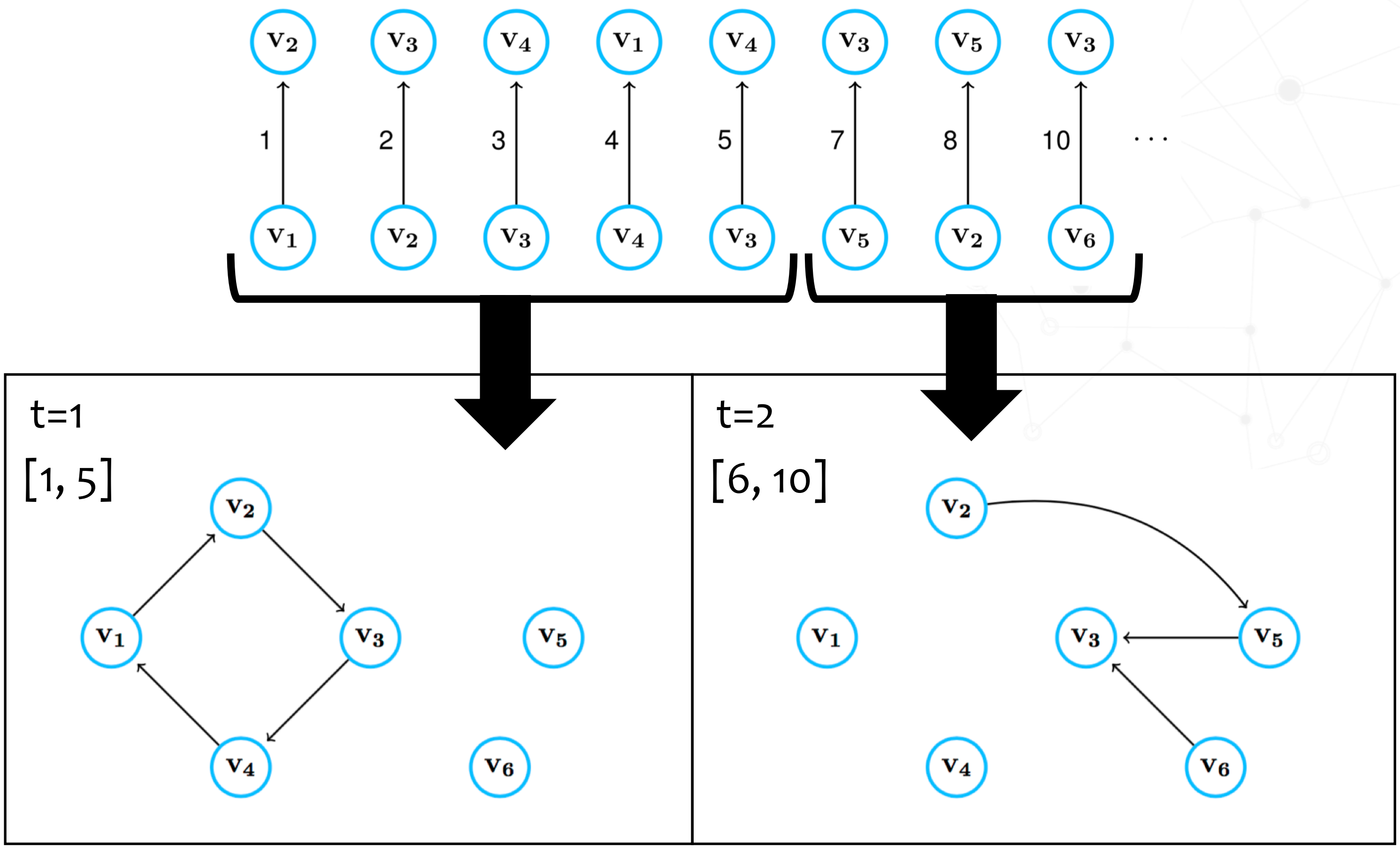


$$e_i = (u, v, t) \in E_T$$

Edge stream

Discrete-time models

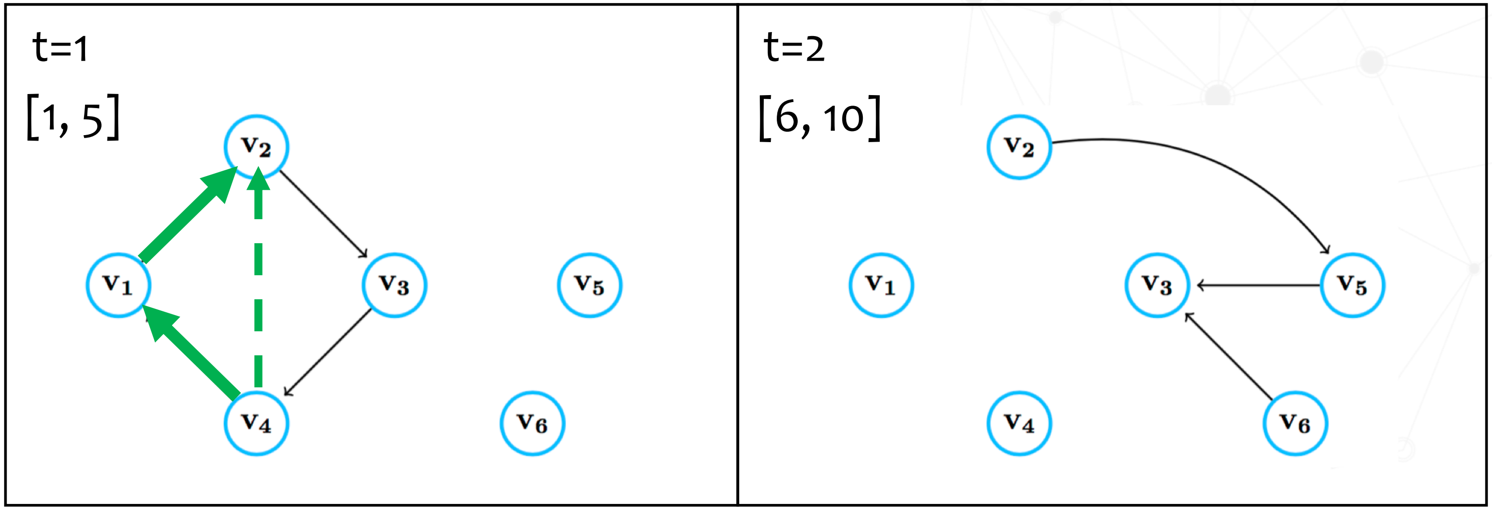
Very coarse approximation of the actual CTDN – temporally invalid & noise/error problems



Discrete-time models: represent dynamic network as a sequence of static snapshot graphs $G_1, \ldots, G_T$ where $G_i = (V, E_t)$
User-defined aggregation time-interval $[t_{i-1}, t_i]$

Discrete-time models

Very coarse approximation of the actual CTDN – temporally invalid & noise/error problems

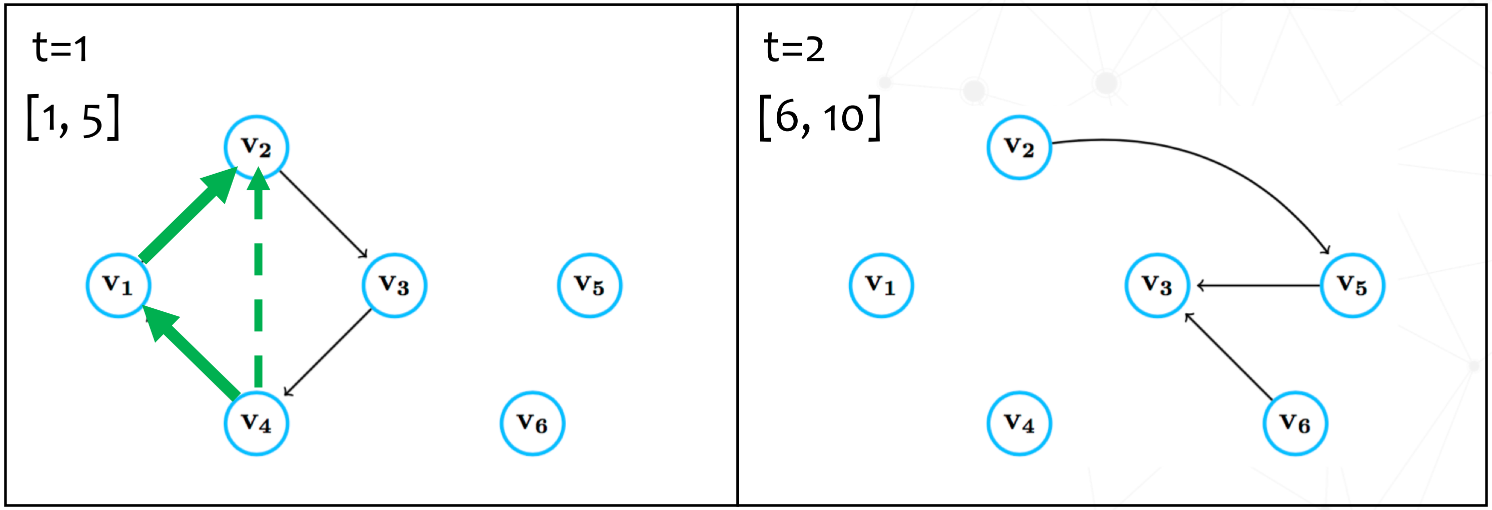


Discrete-time models: represent dynamic network as a sequence of static snapshot graphs $G_1, \ldots, G_T$ where $G_i = (V, E_t)$

A *temporal walk* is a sequence of edges/nodes that obey time.

Discrete-time models

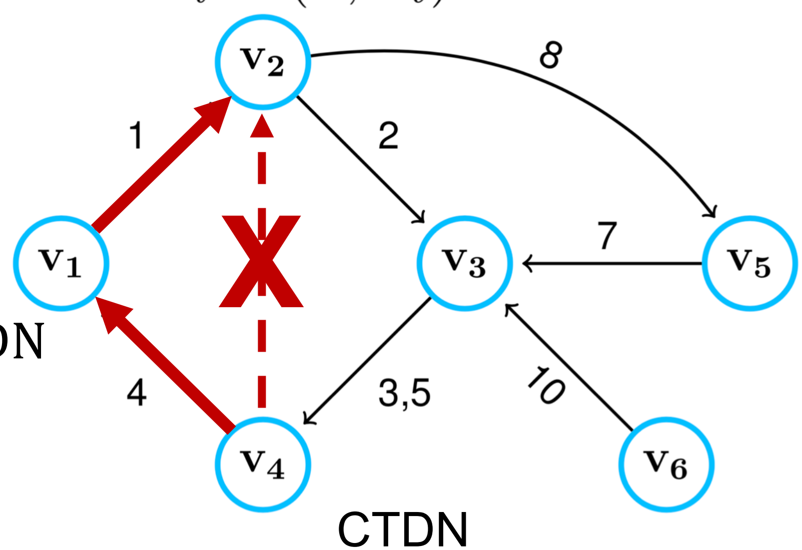
Very coarse representation with similar noise/error problems



Discrete-time models: represent dynamic network as a sequence of static snapshot graphs $G_1, \ldots, G_T$ where $G_i = (V, E_t)$

Notice the walk (v_4, v_1, v_2) is possible despite it being **temporally invalid**

- (v_1, v_2) exists in the past w.r.t. (v_4, v_1)
- No noise/error when modeled as CTDN
- CTDN captures the **temporally valid** walks (with no information loss)



Continuous-Time Dynamic Network Embeddings

- Captures the temporally valid interactions in the dynamic network in a lossless fashion
- **CTDNE's** are **temporally valid embeddings** learned from the actual dynamic network at the finest temporal granularity, e.g., milliseconds
- **CTDNE's** do not have the issues and information loss that arises when the actual dynamic network is *approximated* as a sequence of static snapshot graphs

CTDN Embedding Framework

- Introduces the notion of **temporal walks**
- Serves as a general & unifying framework
  - Existing and future embedding methods that use random walks can be adapted for modeling CTDN’s in a straightforward manner
- Consists of a few interchangeable components

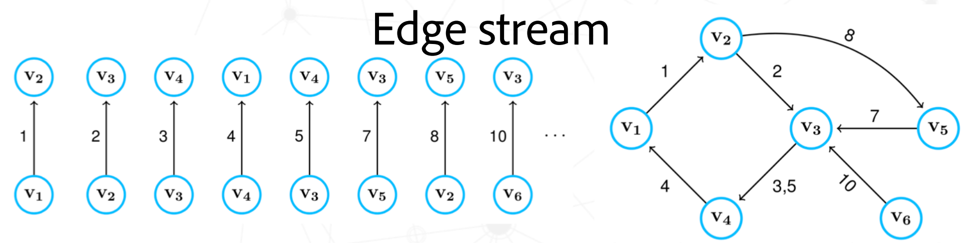
Bias approach to leverage more recent information

Two main ways:

1. Bias the selection of the initial edge to start the temporal random walk
2. Bias the temporal random walk

CTDN Embedding Framework

1. Model network as CTDN



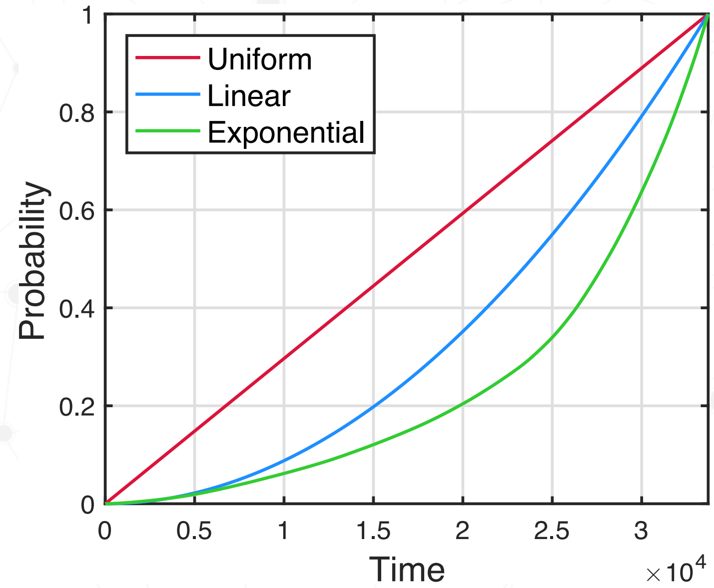
<u>Unbiased/biased Temporal Random Walks</u>

2. Initial temporal edge selection
    - Use **temporally unbiased** or **biased** techniques to sample the initial edge in the temporal walk

3. Temporal neighbor sampling
    - **Temporally unbiased** or **biased** sampling of a node from a temporal neighborhood

4. Learn time-dependent embedding

Initial Temporal Edge Selection



- Each temporal walk starts from a temporal edge $e_i \in E_T$ at time $t = T$ sampled from a distribution $\mathbb{F}_s$

Unbiased $$\mathbb{P}(e) = 1/|E_T|$$

Temporally Biased

- Exponential: $$\mathbb{P}(e) = \frac{\exp\left[\mathcal{T}(e) - t_{\min}\right]}{\sum_{e' \in E_T} \exp\left[\mathcal{T}(e') - t_{\min}\right]}$$

$t_{\min}$ = min. time associated with an edge in G

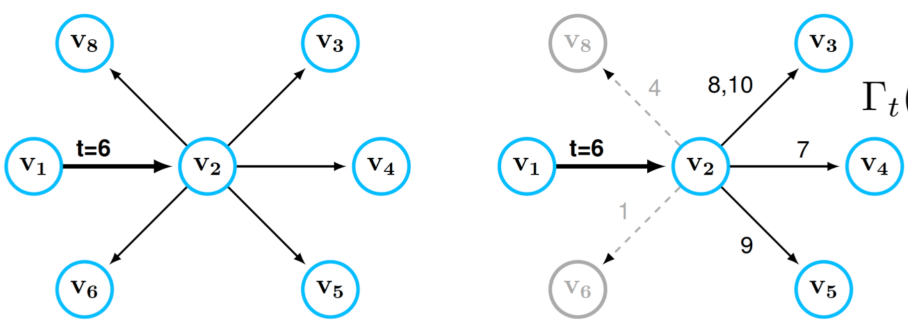
- Linear: $$\mathbb{P}(e) = \frac{\eta(e)}{\sum_{e' \in E_T} \eta(e')}$$ $\eta : E_T \to \mathbb{Z}^+$

Temporal Random Walks

procedure TEMPORALWALK(G', $e = (s, r)$, t, L, C)
  Initialize temporal walk $S_t = [\, s, r \,]$
  Set $i = r$
  for $p = 1$ **to** $min(L, C) - 1$ **do**
    $\Gamma_t(i) = \{(w, t') \mid e = (i, w, t') \in E_T \wedge \mathcal{T}(i) > t\}$
    if $|\Gamma_t(i)| > 0$ **then**
      Select node j from distribution $\mathbb{F}_\Gamma(\Gamma_t(i))$
      Append j to S_t
      Set $t = \mathcal{T}(i, j)$
      Set $i = j$
    else terminate temporal walk
  return temporal walk S_t of length $|S_t|$ rooted at node s

A temporal walk is a temporally valid sequence of edges traversed in increasing order of edge times

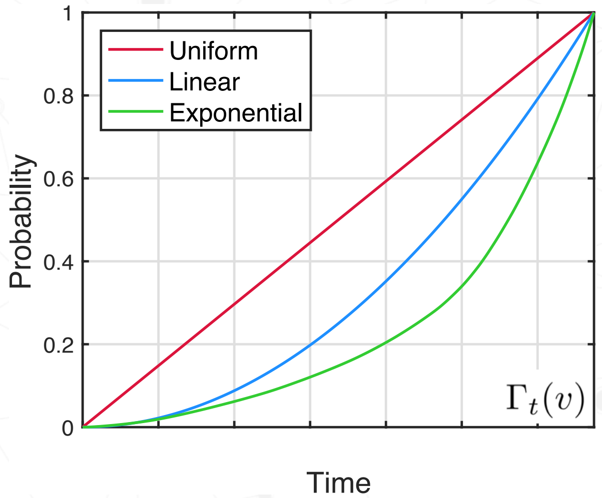
- After sampling the initial edge to begin the temporal walk
- At each step in the temporal random walk, we sample a node w from the temporal neighborhood of node v according to a distribution $\mathbb{F}_\Gamma$
- Afterwards, we add w to the temporal walk, and find the temporal neighbors of w given the edge traversal time, and repeat.



(a) Neighborhood $\Gamma(v_2)$ (b) Temporal neigh. $\Gamma_t(v_2)$

$$\Gamma_t(v) = \{(w, t') \mid e = (v, w, t') \in E_T \wedge \mathcal{T}(e) > t\}$$

Temporal Random Walks



Proceed by sampling a node w from the temporal neighborhood of v, adding it to the temporal walk, traversing (v,w,t), and repeating…

Unbiased $$\mathbb{P}(w) = 1/|\Gamma_t(v)|$$

Temporally Biased

- Exponential: $$\mathbb{P}(w) = \frac{\exp\left[\tau(w) - \tau(v)\right]}{\sum_{w' \in \Gamma_t(v)} \exp\left[\tau(w') - \tau(v)\right]}$$

  $\tau(w)$ = T(v,w)

- Linear: $$\mathbb{P}(w) = \frac{\delta(w)}{\sum_{w' \in \Gamma_t(v)} \delta(w')} \qquad \delta : V \times \mathbb{R}^+ \to \mathbb{Z}^+$$

CTDN Embeddings

Given a temporal walk S_t, we learn time-dependent node embeddings by solving:

$$\max_f \log \mathbb{P}\big(W_T = \{v_{i-\omega}, \cdots, v_{i+\omega}\} \setminus v_i \mid f(v_i)\big)$$

where $f : V \rightarrow \mathbb{R}^D$ is the node embedding function; and

$$W_T = \{v_{i-\omega}, \cdots, v_{i+\omega}\} \text{ s.t.}$$

$$\mathcal{T}(v_{i-\omega}, v_{i-\omega+1}) < \cdots < \mathcal{T}(v_{i+\omega-1}, v_{i+\omega})$$

is an arbitrary temporal context window $W_T \subseteq S_t$

Just one example extending the Skip-Gram model,
many other possibilities

Experiments

Experiments

Use first 75% of edges (ordered by time) as training & last 25% for testing. We sample an equal number of negative edges to use. (more details in paper)

CTDNE: $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$ = uniform (simplest)

DeepWalk & D=128, R=10, L=80, ω=10

Node2vec($p, q \in \{0.25, 0.50, 1, 2, 4\}$)

LINE: 2nd-order, samples T = 60M

$$\beta = \underbrace{R \times N}_{\text{\# of total walks}} \times \underbrace{(L - \omega + 1)}_{\text{\# of context windows from walk of length L}}$$

Table 1: AUC scores for Temporal Link Prediction.

DATA	DeepWalk	Node2Vec	LINE	CTDNE	(GAIN)
ia-contact	0.845	0.874	0.736	**0.913**	(+10.37%)
ia-hypertext09	0.620	0.641	0.621	**0.671**	(+6.51%)
ia-enron-employees	0.719	0.759	0.550	**0.777**	(+13.00%)
ia-radoslaw-email	0.734	0.741	0.615	**0.811**	(+14.83%)
ia-email-eu	0.820	0.860	0.650	**0.890**	(+12.73%)
fb-forum	0.670	0.790	0.640	**0.826**	(+15.25%)
soc-bitcoinA	0.840	0.870	0.670	**0.891**	(+10.96%)
soc-wiki-elec	0.820	0.840	0.620	**0.857**	(+11.32%)

repeated for 10 random trials

Overall gain in AUC of 11.9% across all embedding methods and graphs

Experiments comparing different CTDNE variants

$\mathbb{F}_s$ = distribution used to select the initial edge to begin a temporal walk

$\mathbb{F}_\Gamma$ = distribution used to select next "temporally relevant node" in a temporal walk

Variant $\mathbb{F}_s$	$\mathbb{F}_\Gamma$	contact	hyper	enron	rado
Unif (Eq. 1)	**Unif** (Eq. 5)	0.913	0.671	0.777	0.811
Unif (Eq. 1)	**Lin** (Eq. 7)	0.903	0.665	0.769	0.797
Lin (Eq. 3)	**Unif** (Eq. 5)	0.915	0.675	0.773	0.818
Lin (Eq. 3)	**Lin** (Eq. 7)	0.903	0.667	0.782	0.806
Exp (Eq. 2)	**Exp** (Eq. 6)	**0.921**	0.681	**0.800**	0.820
Unif (Eq. 1)	**Exp** (Eq. 6)	0.913	0.670	0.759	0.803
Exp (Eq. 2)	**Unif** (Eq. 5)	0.920	**0.718**	0.786	**0.827**
Lin (Eq. 3)	**Exp** (Eq. 6)	0.916	0.681	0.782	0.823
Exp (Eq. 2)	**Lin** (Eq. 7)	0.914	0.675	0.747	0.817

Results indicate the choice of distribution depends on the underlying data and temporal characteristics.

Comparing CTDNE's to DTDNE's (discrete static snapshot approaches)

Two types of embedding methods:

- Discrete-time dynamic network embeddings (DTDNE)
- Continuous-time dynamic network embeddings (CTDNE)

DTDNE methods: Given T static snapshot graphs, we learn a (D/T)-dimensional embedding and concatenate them all to obtain a D-dimensional embedding

Disadvantages/limitations:

- **Approximate & noisy representation**
- **Uses temporally invalid info.**
- **Finding appropriate aggregation granularity is NP-hard**
  - Heuristics often used or simply ignored
- **How to handle inactive nodes? Many heuristics…**
  - Use previous embedding (if exists)
  - Set to mean embbedding
  - Set to zero, etc…

Comparing CTDNE's to DTDNE's (discrete static snapshot approaches)

Two types of embedding methods:

- Discrete-time dynamic network embeddings (DTDNE)
- Continuous-time dynamic network embeddings (CTDNE)

DTDNE methods: Given T static snapshot graphs, we learn a (D/T)-dimensional embedding and concatenate them all to obtain a D-dimensional embedding

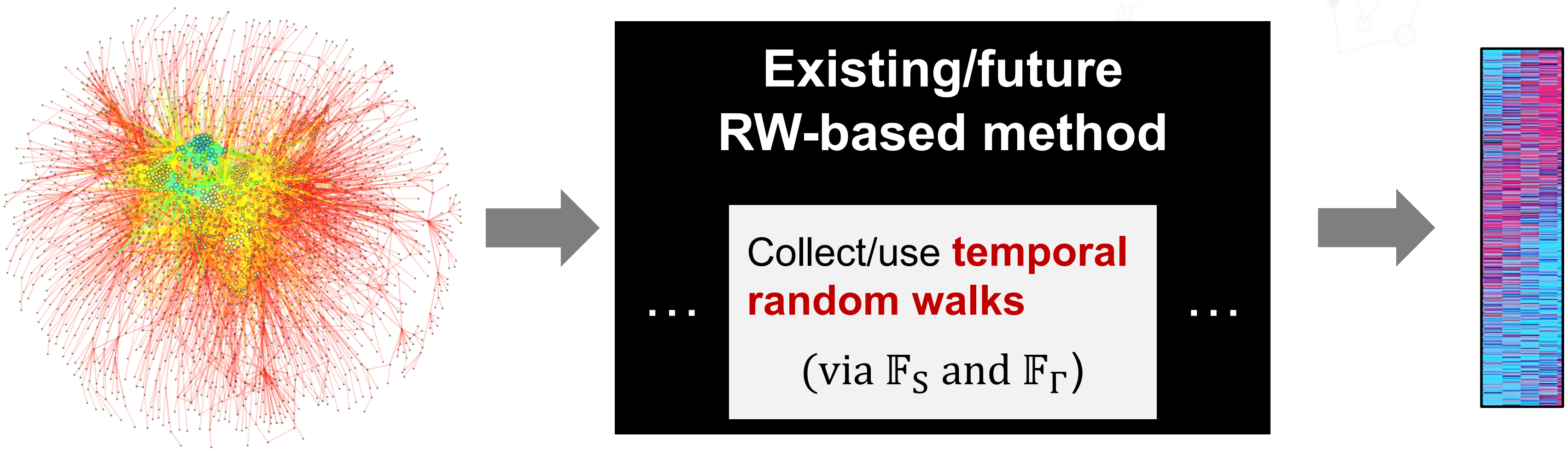
Results comparing CTDNE's to DTDNE's (AUC)

DATA	**DTDNE**	**CTDNE**	(GAIN)
ia-contact	0.843	**0.913**	(+8.30%)
ia-hypertext09	0.612	**0.671**	(+9.64%)
ia-enron-employees	0.721	**0.777**	(+7.76%)
ia-radoslaw-email	0.785	**0.811**	(+3.31%)

Overall, CTDN embeddings capture the temporal properties better & more accurately than embedding methods that use a sequence of discrete snapshot graphs (and without all the issues/heuristics)

CTDN Embedding Framework

- This work learns CTDNE's using basic Skip-gram model
- Other existing or future RW-based embedding methods can be easily generalized via the proposed framework



Examples: node2vec, struct2vec, and deep graph models, e.g., GRAM

Summary and Conclusion

- Introduced the notion of temporal random walks for embedding methods
- Continuous-Time Dynamic Network Embeddings
  - Avoids the issues and loss in information from ignoring time or creating discrete static snapshot graphs
- General & Unifying Framework
  - Key idea can be used by others to adapt existing and/or future embedding methods in a straightforward way
- Effectiveness
  - Achieves an average gain in AUC of 11.9% across all methods and graphs from various application domains

Thanks!

Questions?

Data accessible online:

http://networkrepository.com

References

- Soundarajan, S., Tamersoy, A., Khalil, E. B., Eliassi-Rad, T., Chau, D. H., Gallagher, B., & Roundy, K. (2016, April). Generating graph snapshots from streaming edge data. In *Proceedings of the 25th International Conference Companion on World Wide Web* (pp. 109-110).
- Ryohei Hisano. Semi-supervised graph em- bedding approach to dynamic link prediction. arXiv preprint arXiv:1610.04351, 2016.
- Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. arXiv preprint, arXiv:1710.10368, 2017