From random-walks to graph-sprints: a low-latency node embedding framework on continuous-time dynamic graphs

Ahmad Naser Eddin, Jacopo Bono, David Aparício, Hugo Ferreira, João Ascensão, Pedro Ribeiro, Pedro Bizarro

Contributions
We propose **Graph-sprints**, a graph feature extraction framework which computes node embeddings in the form of histograms characterizing a node’s neighborhood in dynamic graphs.

Our contributions include:
- A streaming, low-latency graph feature engineering method for continuous time dynamic graphs.
- Benchmarking Graph-sprints against state-of-the-art methods using five different datasets.

We show that Graph-sprints features, combined with a neural network classifier, are up to 10x faster to run while achieving up to +5% AUC in binary node classification compared with the higher-latency GNNs.

Motivation
- Typical GNNs are computationally heavy, resulting in high inference latencies.
- Performing the graph aggregations asynchronously results in inference using outdated information.

Our aim is to design a system that:
- Enables low-latency inference.
- Uses most up-to-date information during the embedding calculation.
- Is competitive with state-of-the-art, higher latency methods.

Methods

(A) **Temporal Random Walk**

(B) **Unfolded Walk**

(C) **Naive Node Embedding**

(D) **Streaming Node Embedding**

Embed a node's old embedding $s_0$, the interacting node’s embedding $s_1$ and a new edge features $f_0$, a new node embedding is calculated using the following formula:

$$
\tilde{s}_0 \leftarrow \beta \tilde{s}_0 + (1 - \beta) \left( (1 - \alpha) \tilde{s}_1 + \alpha s_1 \right)
$$

The parameters $\alpha$ and $\beta$ control how quickly older information is forgotten.

Results

Node classification AUC vs Inference runtime

A neural network classifier that uses **Graph-sprints (GS)** or **Graph-sprints + raw (GS+raw)** features achieves the best performance in three node classification tasks, compared to state-of-the-art methods (Jodie[1], TGN[2]).

Graph-sprints (GS) is considerably faster to run. Moreover, unlike other methods, GS inference time remains constant when the number of edges increase (larger/denser graph). Thus the speedups obtained increase with the number of edges in the graph.

Similar results were obtained in the two internal datasets (see article).

References