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.

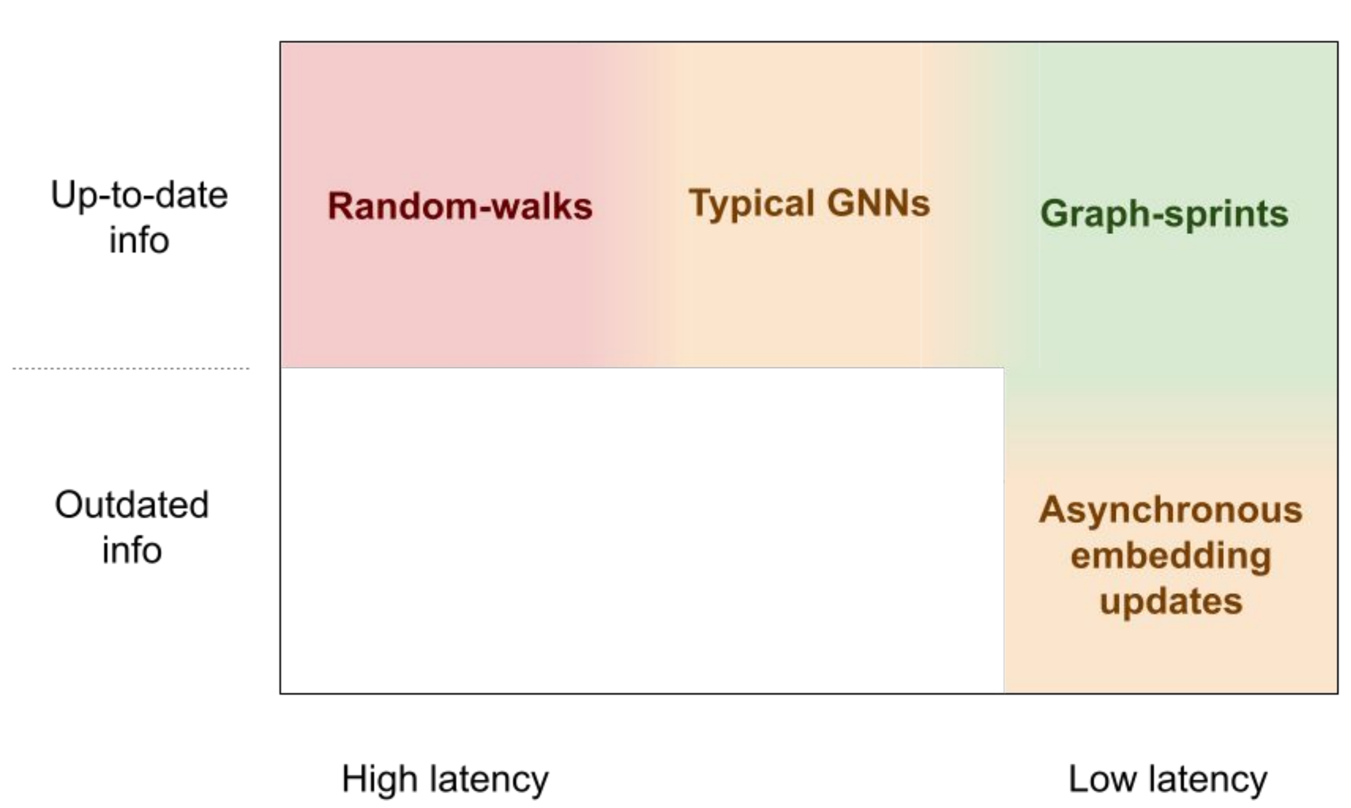
We show that Graph-sprints features, combined with a neural network classifier, are **up to 10x faster to run** while achieving **up to +5% AUC** in binary node classification compared with the higher-latency GNNs.

Motivation

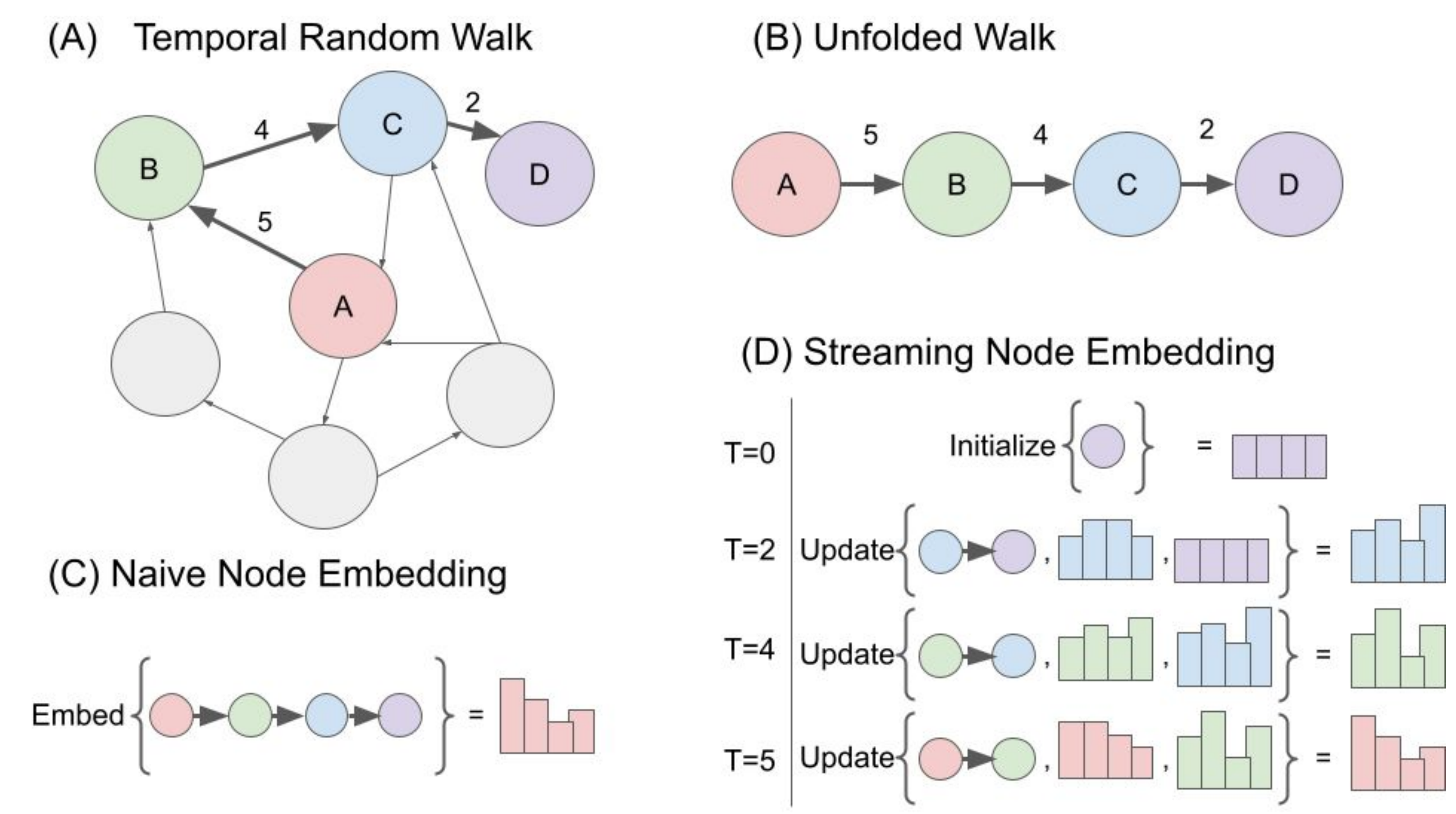
- Typical GNNs are **computationally heavy**, resulting in high inference latencies.
- Performing the graph aggregations asynchronously results in inference using **outdated information.**

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. A temporal random-walk is traversed from the most recent interaction A-B towards older interactions.
B. The same random-walk can be seen as a timeseries of edges.
C. A full temporal random-walk can be summarized by aggregating encountered feature values.
D. One can compute similar summaries in a streaming setting, from only the new edge and the existing embeddings of the involved nodes.

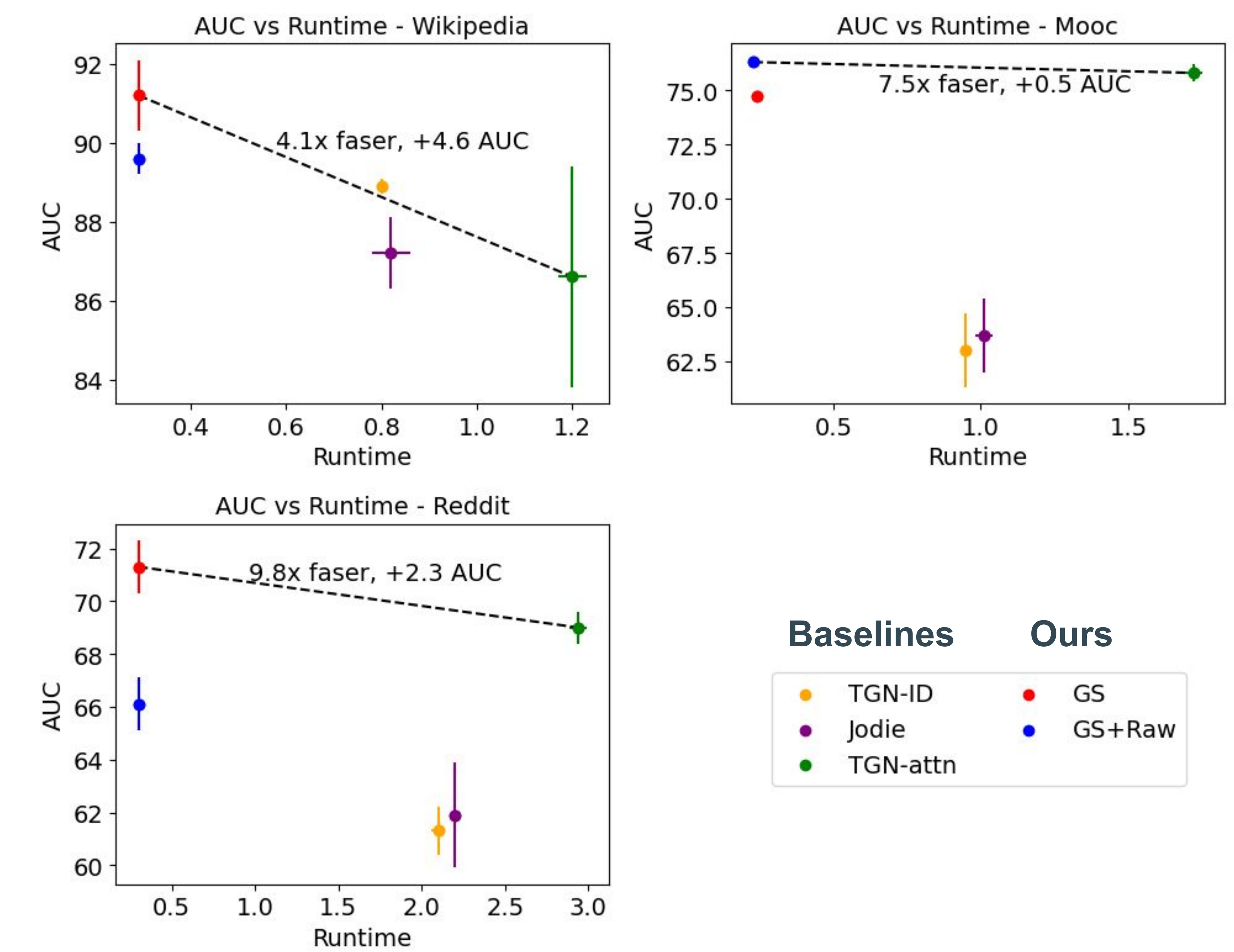
Step **D** illustrates Graph-sprints. Formally, given a node's old embedding s_0, the interacting node's embedding s_1 and a new edge features f_0, a new node embedding is calculated using the following formula:

$$\vec{s}_0 \leftarrow \beta\vec{s}_0 + (1-\beta)\left((1-\alpha)\vec{\delta}(f_0) + \alpha\vec{s}_1\right)$$

The parameters α and β 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 at inference. 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

[1] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM.
[2] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In ICML 2020 Workshop on Graph Representation Learning.