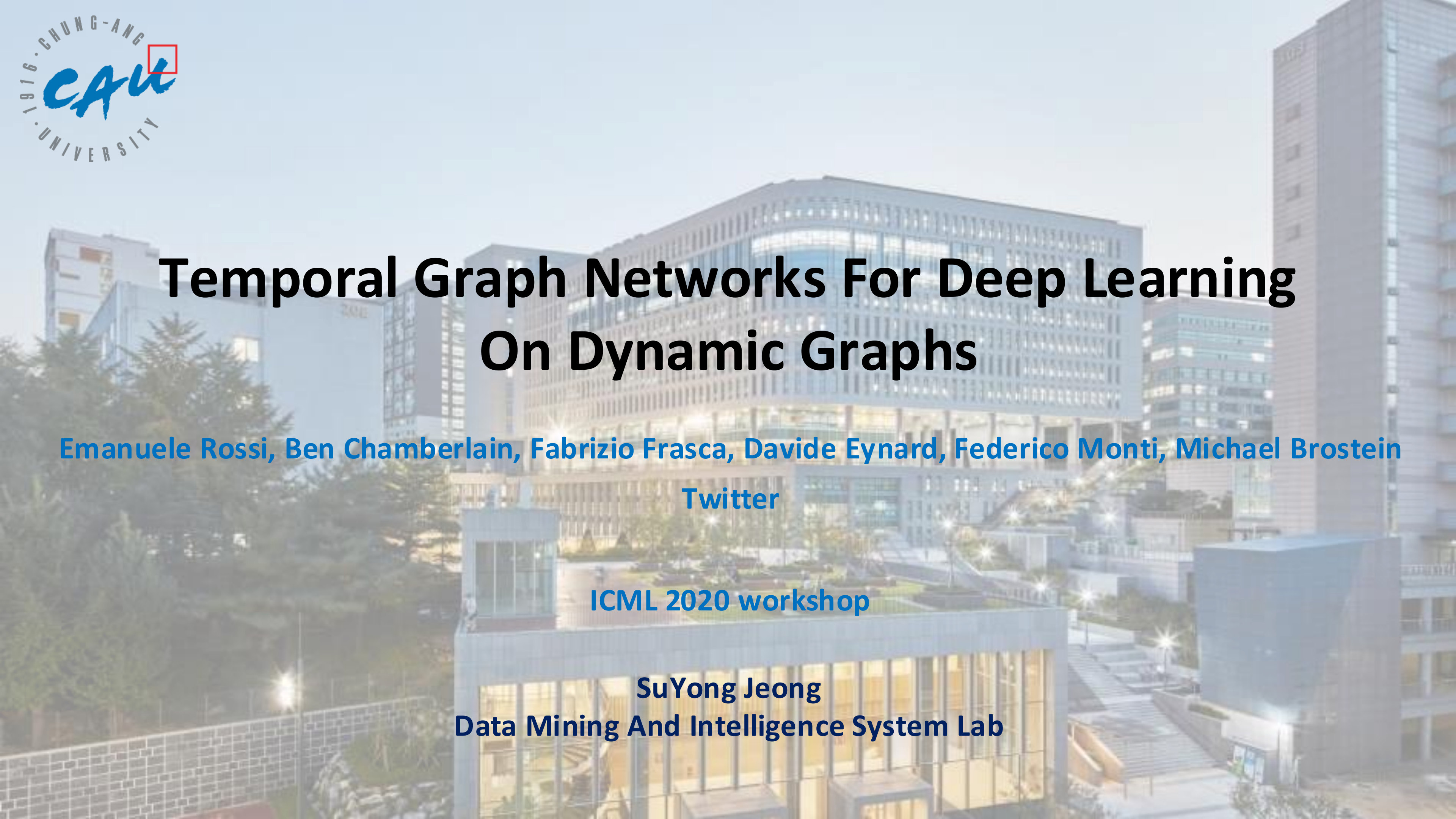


Temporal Graph Networks For Deep Learning On Dynamic Graphs

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, Michael Brostein

Twitter

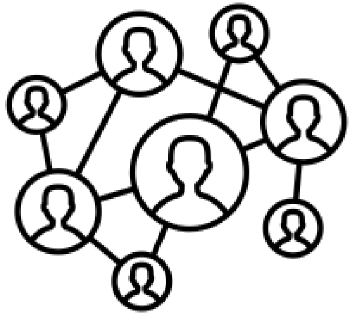
ICML 2020 workshop

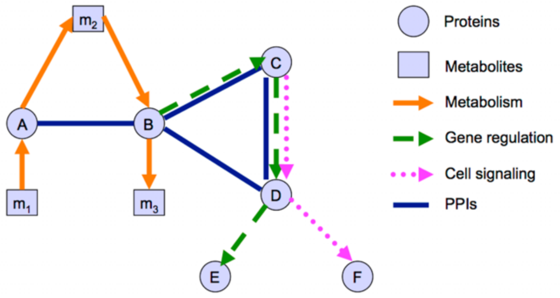
SuYong Jeong
Data Mining And Intelligence System Lab

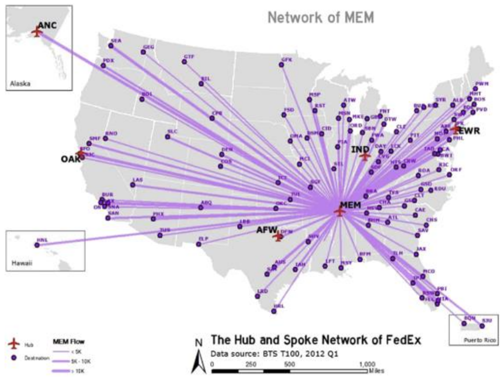
Outline

Background

- Graphs Can Represent Everything
  - With nodes and edges
  - Social network, biology, hub & spoke network, etc.
  - Enable the discovery of diverse information in data

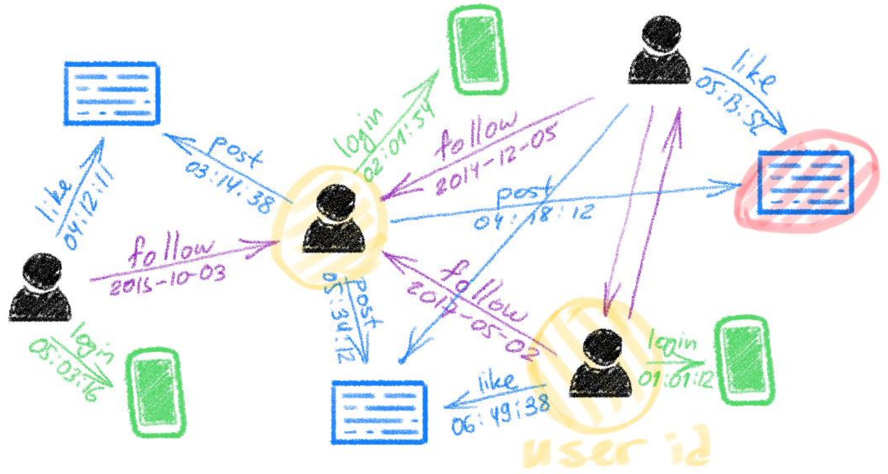


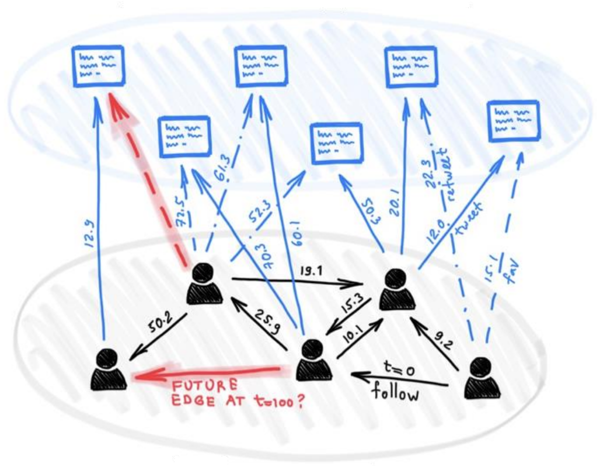




Background

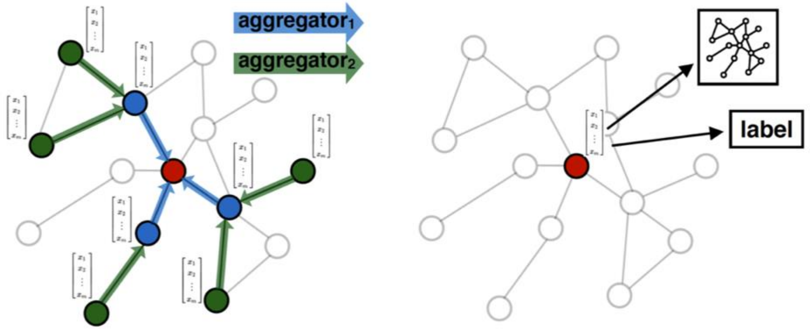
- Characteristics of Real-World
  - Mostly **dynamic**, not static, **evolving over time**
  - Crucial insights can be contained in the dynamic structure



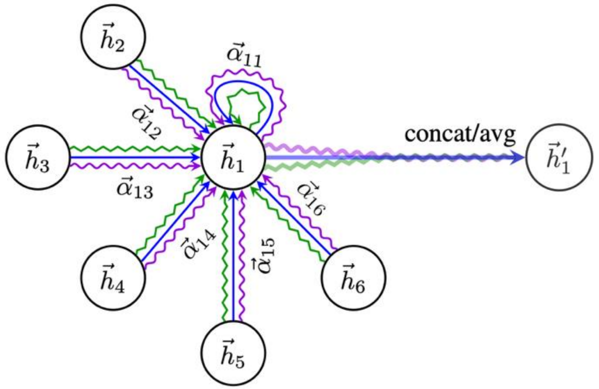


Problem

- Static GNNs
  - Assume the underlying graph is static
    - GCN, GraphSAGE, GAT, etc.
  - Hard to represent real-world characteristic
    - Only using the final state as a static snapshot of the graph



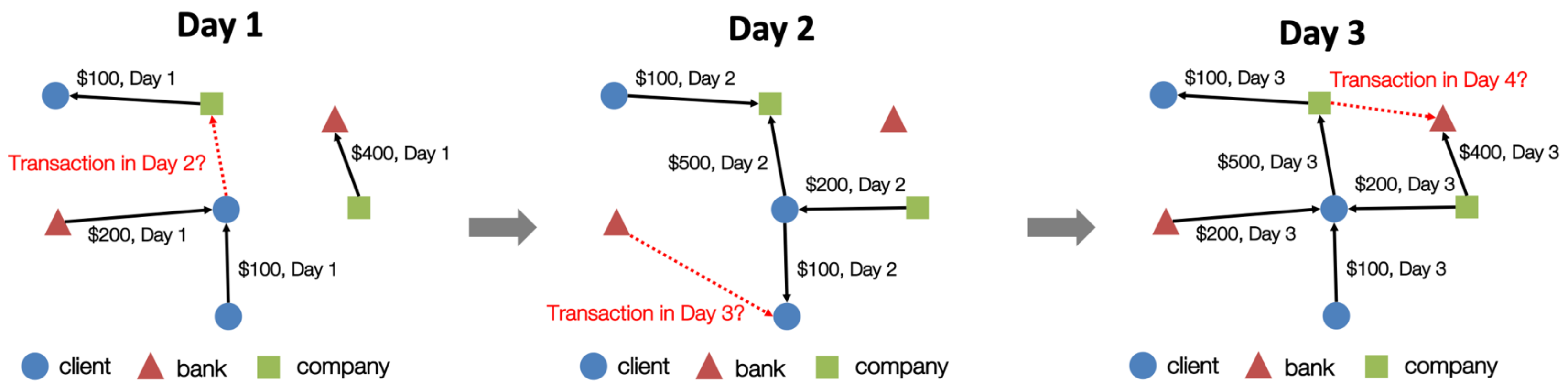
GraphSAGE



GAT

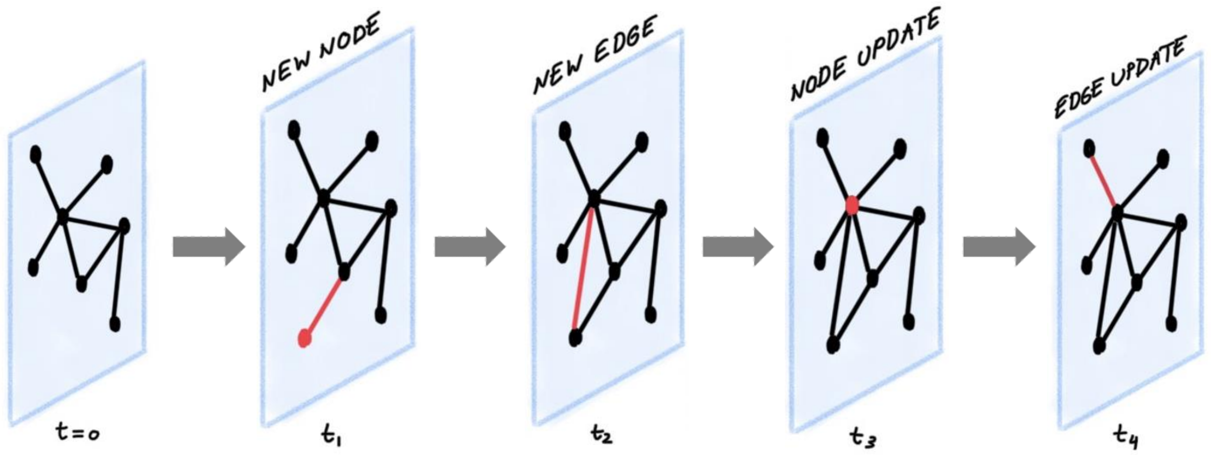
Methodology

- Discrete-Time Dynamic Graphs
  - Sequences of static graph snapshots taken at intervals in time
  - Used to represent coarse-grained, macro-level changes
    - Unsuitable for some real-world settings



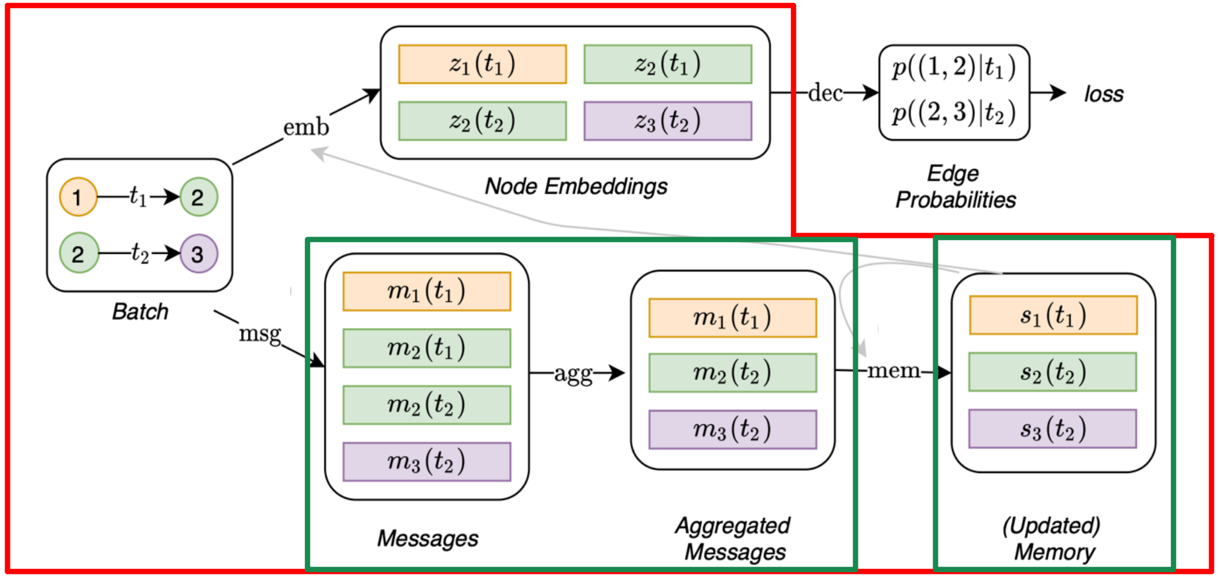
Methodology

- Continuous-Time Dynamic Graphs
  - Represented as timed lists of events
    - Node-wise events
    - Edge-wise events
  - Used to represent fine-grained changes
    - Well-suited for real-time tasks



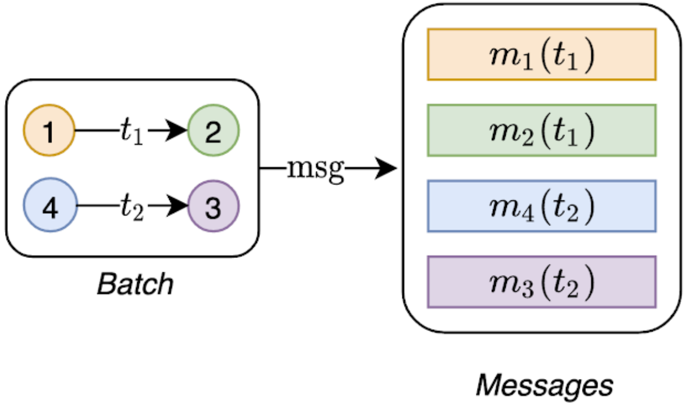
Methodology

- Temporal Graph Network
  - A novel encoder applied on CTDGs
  - Message function
  - Memory module



TGN - Components

- Message Functions on TGN
  - Computing messages for the event
  - Aggregation of multiple messages into a single one
  - The messages are used to update the memory



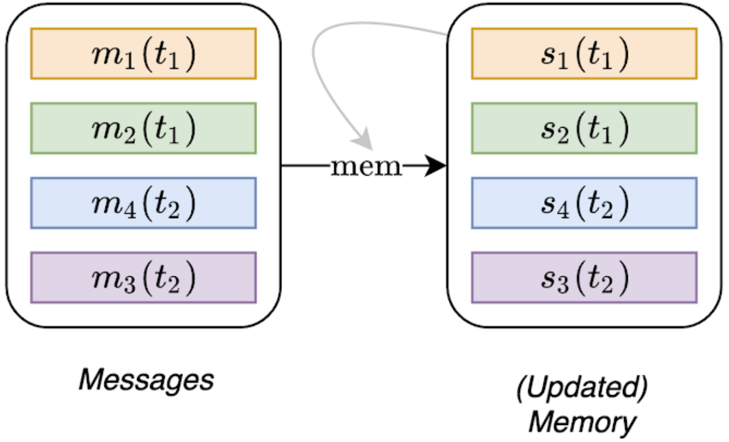
$$\mathbf{m}_i(t) = \mathrm{msg}\left(\mathbf{s}_i(t^-), \mathbf{s}_j(t^-), t, \mathbf{e}_{ij}(t)\right)$$

$$\mathbf{m}_j(t) = \mathrm{msg}\left(\mathbf{s}_j(t^-), \mathbf{s}_i(t^-), t, \mathbf{e}_{ij}(t)\right)$$

$$\bar{\mathbf{m}}_i(t) = \mathrm{agg}\left(\mathbf{m}_i(t_1), \ldots, \mathbf{m}_i(t_b)\right).$$

TGN - Components

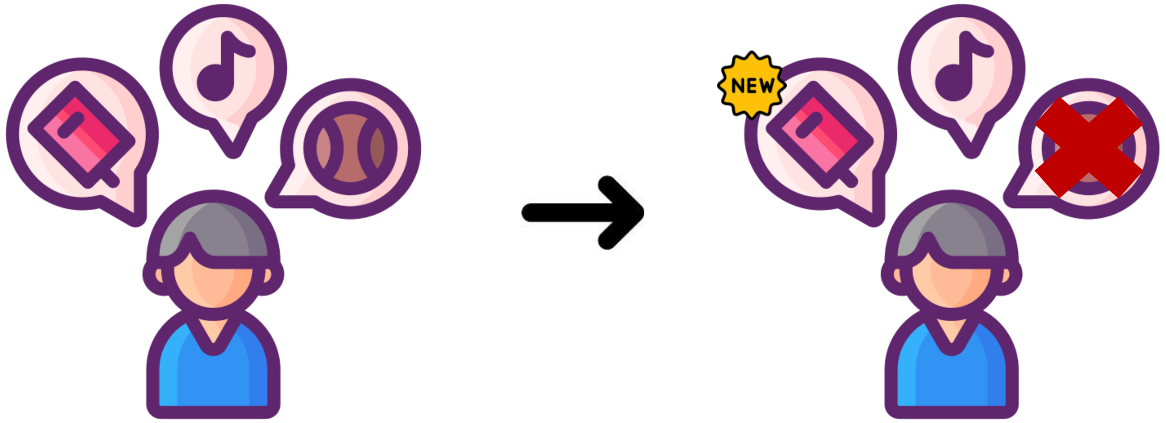
- Memory Modules on TGN
  - Storing the states of all the nodes
  - Representing **the node's history** in a compressed format
  - Updated after an event with the new messages



$$\mathbf{s}_i(t) = \mathrm{mem}\left(\mathbf{m}_i(t), \mathbf{s}_i(t^-)\right)$$

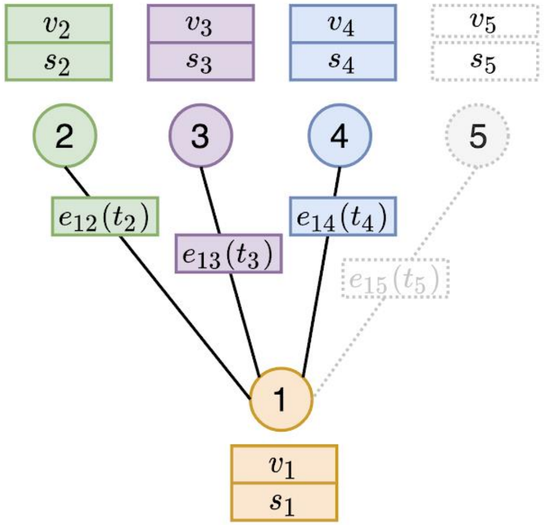
TGN - Node Embedding

- Directly Embedding Using Node’s Memory
  - A bad idea due to the **staleness problem**
  - The node with long period of inactivity, its memory goes out of date
    - Imagine returning to social media after a long break



TGN - Node Embedding

- Aggregation to Solve the Staleness Problem
  - Some of neighbors have been active
  - By aggregating their memories, TGN can compute up-to-date embeddings

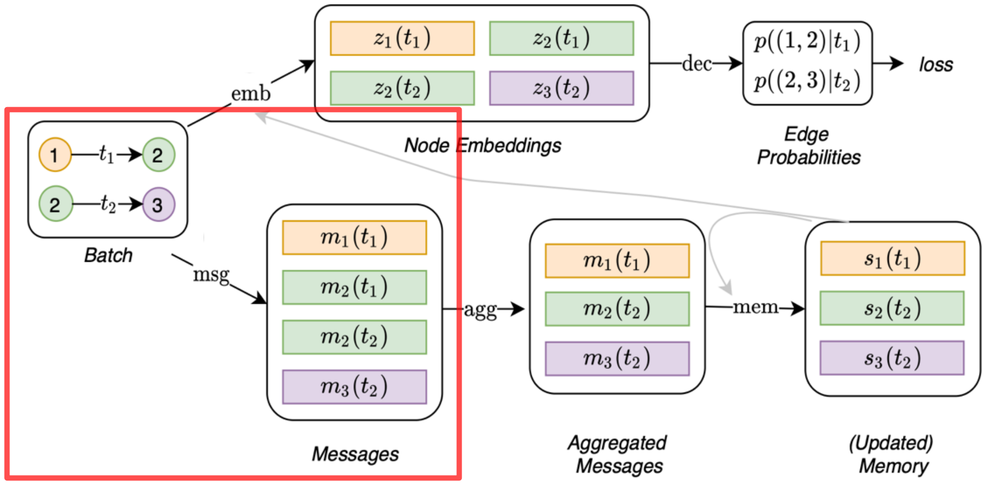


$$\mathbf{z}_i(t) = \mathrm{emb}(i,t) = \sum_{j \in n_i^k([0,t])} h\left(\mathbf{s}_i(t), \mathbf{s}_j(t), \mathbf{e}_{ij}, \mathbf{v}_i(t), \mathbf{v}_j(t)\right)$$

$$\begin{aligned}
\mathbf{h}_i^{(l)}(t) &= \mathrm{MLP}^{(l)}(\mathbf{h}_i^{(l-1)}(t) \,\|\, \tilde{\mathbf{h}}_i^{(l)}(t)), \\
\tilde{\mathbf{h}}_i^{(l)}(t) &= \mathrm{MultiHeadAttention}^{(l)}(\mathbf{q}^{(l)}(t), \mathbf{K}^{(l)}(t), \mathbf{V}^{(l)}(t)), \\
\mathbf{q}^{(l)}(t) &= \mathbf{h}_i^{(l-1)}(t) \,\|\, \boldsymbol{\phi}(0), \\
\mathbf{K}^{(l)}(t) &= \mathbf{V}^{(l)}(t) = \mathbf{C}^{(l)}(t), \\
\mathbf{C}^{(l)}(t) &= [\mathbf{h}_1^{(l-1)}(t) \,\|\, \mathbf{e}_{i1}(t_1) \,\|\, \boldsymbol{\phi}(t-t_1), \ldots, \mathbf{h}_N^{(l-1)}(t) \,\|\, \mathbf{e}_{iN}(t_N) \,\|\, \boldsymbol{\phi}(t-t_N)].
\end{aligned}$$

TGN - Training

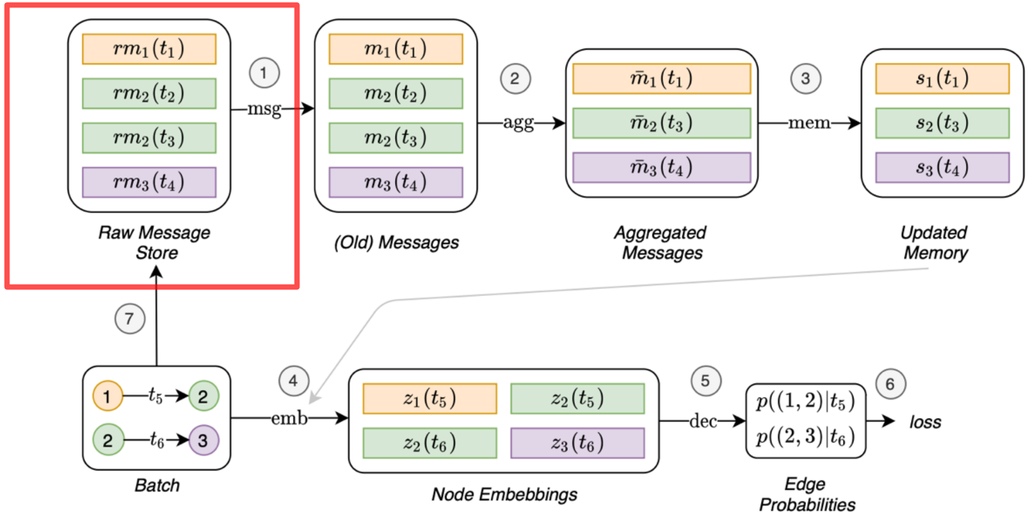
- Overall Process of TGN
  - Making **messages** from events
  - **Updating nodes' memory** using new messages
  - Embedding nodes by **aggregating their memory with its neighbors**



→ Causes an **information leakage**

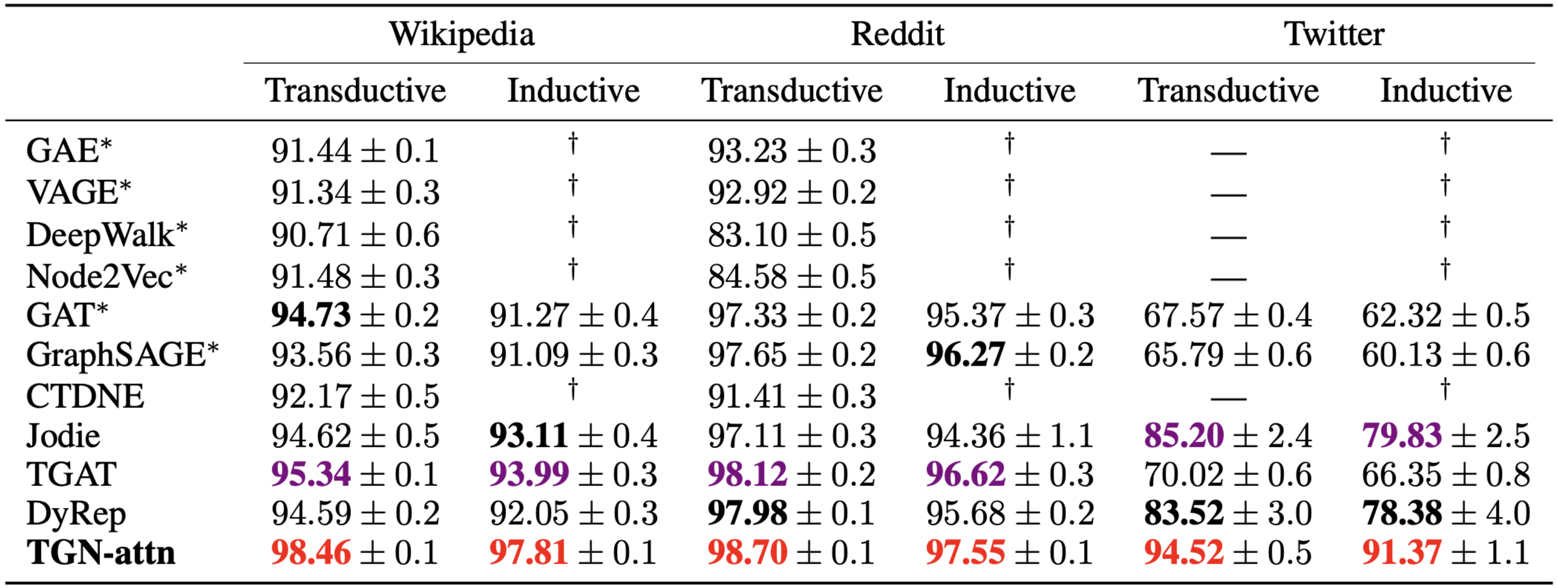
TGN - Training

- To Prevent the Information Leakage
  - Introducing the **raw message store** recording past interactions
  - Updating the memory with messages coming from previous batches



Experiment

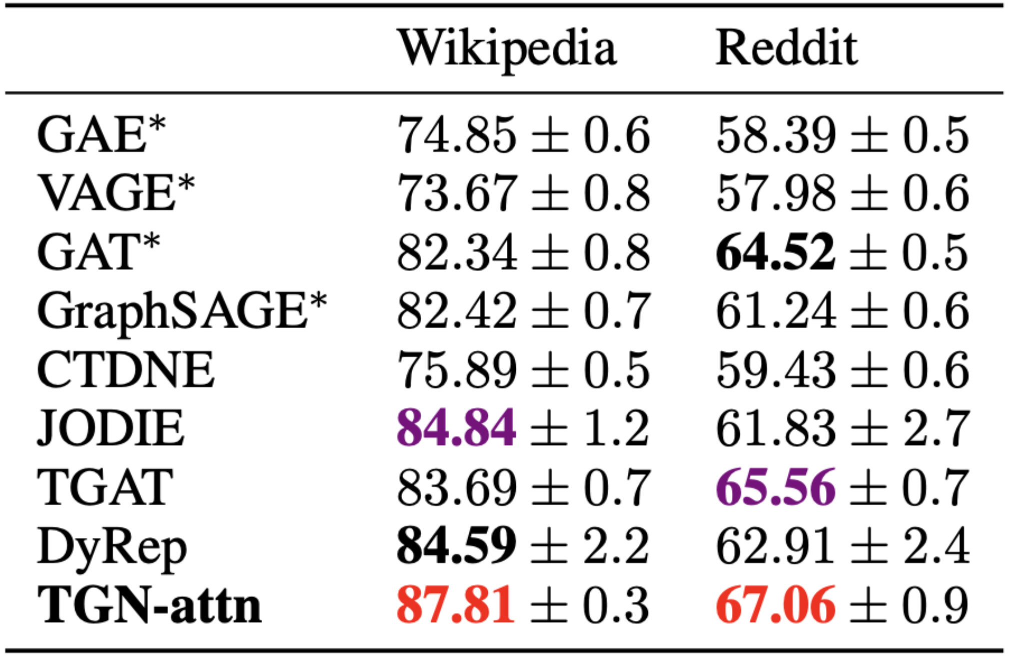
- A Comparison of Various GNNs
  - Future link prediction task in transductive and inductive settings



	Wikipedia		Reddit		Twitter	
	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
GAE*	91.44 ± 0.1	†	93.23 ± 0.3	†	—	†
VAGE*	91.34 ± 0.3	†	92.92 ± 0.2	†	—	†
DeepWalk*	90.71 ± 0.6	†	83.10 ± 0.5	†	—	†
Node2Vec*	91.48 ± 0.3	†	84.58 ± 0.5	†	—	†
GAT*	**94.73** ± 0.2	91.27 ± 0.4	97.33 ± 0.2	95.37 ± 0.3	67.57 ± 0.4	62.32 ± 0.5
GraphSAGE*	93.56 ± 0.3	91.09 ± 0.3	97.65 ± 0.2	**96.27** ± 0.2	65.79 ± 0.6	60.13 ± 0.6
CTDNE	92.17 ± 0.5	†	91.41 ± 0.3	†	—	†
Jodie	94.62 ± 0.5	**93.11** ± 0.4	97.11 ± 0.3	94.36 ± 1.1	**85.20** ± 2.4	**79.83** ± 2.5
TGAT	**95.34** ± 0.1	**93.99** ± 0.3	**98.12** ± 0.2	**96.62** ± 0.3	70.02 ± 0.6	66.35 ± 0.8
DyRep	94.59 ± 0.2	92.05 ± 0.3	**97.98** ± 0.1	95.68 ± 0.2	**83.52** ± 3.0	**78.38** ± 4.0
TGN-attn	**98.46** ± 0.1	**97.81** ± 0.1	**98.70** ± 0.1	**97.55** ± 0.1	**94.52** ± 0.5	**91.37** ± 1.1

Experiment

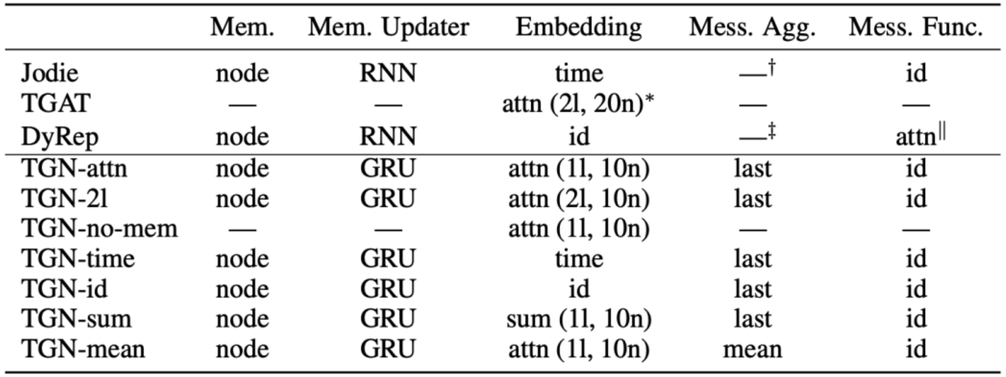
- A Comparison of Various GNNs
  - Dynamic node classification on the same settings



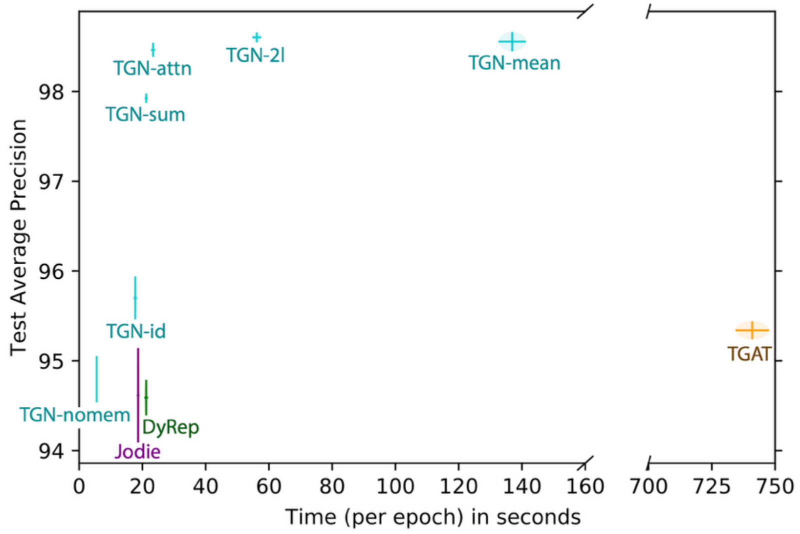
	Wikipedia	Reddit
GAE*	74.85 ± 0.6	58.39 ± 0.5
VAGE*	73.67 ± 0.8	57.98 ± 0.6
GAT*	82.34 ± 0.8	**64.52** ± 0.5
GraphSAGE*	82.42 ± 0.7	61.24 ± 0.6
CTDNE	75.89 ± 0.5	59.43 ± 0.6
JODIE	**84.84** ± 1.2	61.83 ± 2.7
TGAT	83.69 ± 0.7	**65.56** ± 0.7
DyRep	**84.59** ± 2.2	62.91 ± 2.4
TGN-attn	**87.81** ± 0.3	**67.06** ± 0.9

Experiment

- A Comparison of Various Configure of TGN and Older Methods
  - Future link prediction in the transductive setting (Wikipedia)

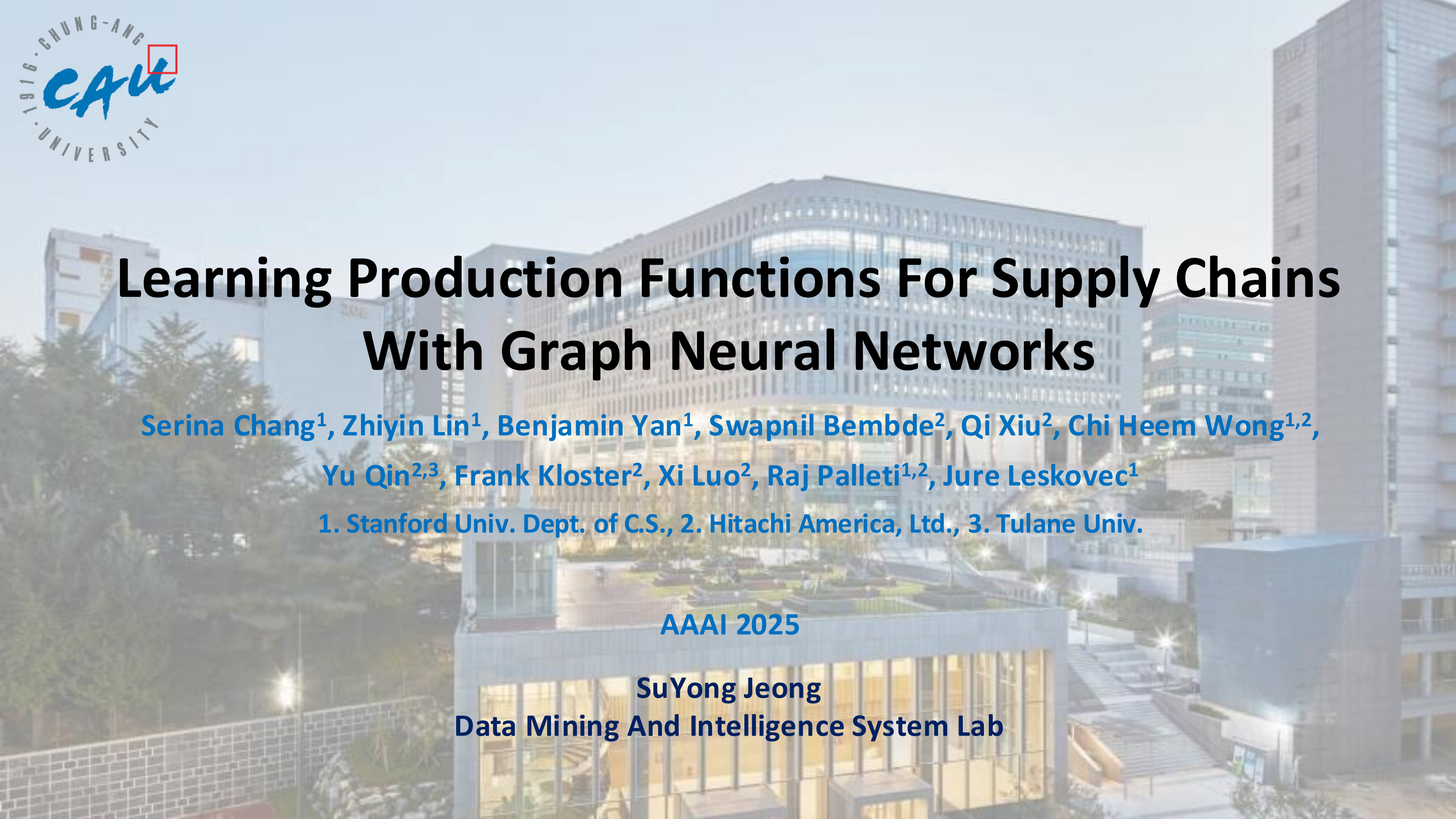


	Mem.	Mem. Updater	Embedding	Mess. Agg.	Mess. Func.
Jodie	node	RNN	time	—†	id
TGAT	—	—	attn (2l, 20n)*	—	—
DyRep	node	RNN	id	—‡	attn‖
TGN-attn	node	GRU	attn (1l, 10n)	last	id
TGN-2l	node	GRU	attn (2l, 10n)	last	id
TGN-no-mem	—	—	attn (1l, 10n)	—	—
TGN-time	node	GRU	time	last	id
TGN-id	node	GRU	id	last	id
TGN-sum	node	GRU	sum (1l, 10n)	last	id
TGN-mean	node	GRU	attn (1l, 10n)	mean	id



Conclusion

- ☐ GNNs based on static graphs have limitations in representing real-world scenarios
- ☐ TGNs can represent continuous-time dynamic graphs that evolve according to events using each node’s memory
- ☐ TGNs generate up-to-date node embeddings through node’s memory and temporal neighborhood aggregation
- ☐ TGNs are effective for capturing fine-grained event patterns in real time



Learning Production Functions For Supply Chains With Graph Neural Networks

Serina Chang[1], Zhiyin Lin[1], Benjamin Yan[1], Swapnil Bembde[2], Qi Xiu[2], Chi Heem Wong[1,2], Yu Qin[2,3], Frank Kloster[2], Xi Luo[2], Raj Palleti[1,2], Jure Leskovec[1]

1. Stanford Univ. Dept. of C.S., 2. Hitachi America, Ltd., 3. Tulane Univ.

AAAI 2025

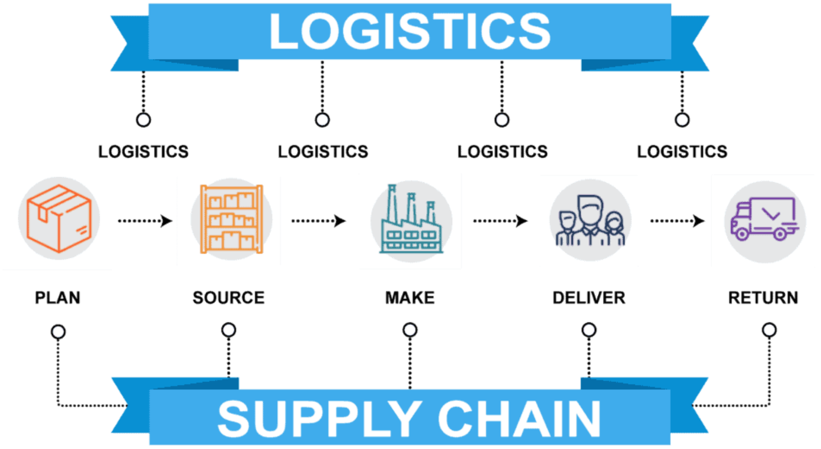
SuYong Jeong
Data Mining And Intelligence System Lab

Outline

- ☐ **Background**
  - ■ Introduction to Supply Chains
- ☐ **Problem**
- ☐ **Methodology**
- ☐ **Experiment**
  - ■ SupplySim
- ☐ **Conclusion**

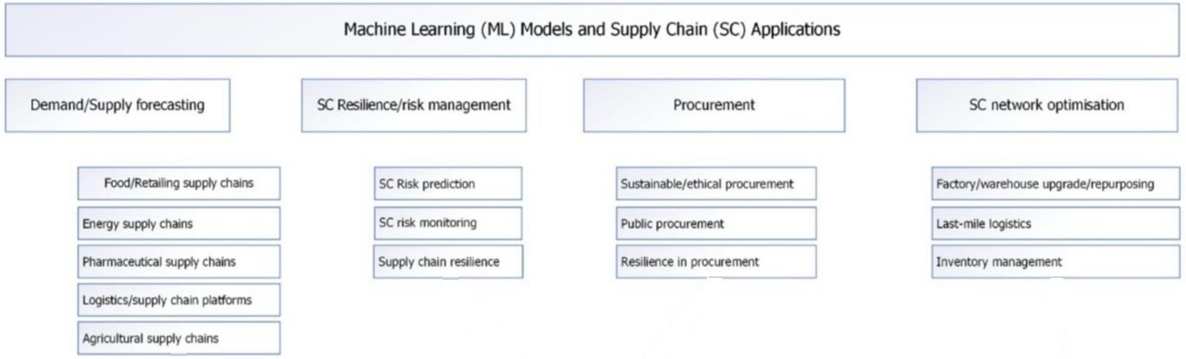
Introduction to Supply Chains

- A Backbone of the Global Economy
  - The network of entities through which material flows
    - Including suppliers, carriers, manufacturing sites, retailers and customers
  - Disruptions of SCs may lead to massive costs and risk national stability
    - Modeling supply chains and how they evolve is essential



Background

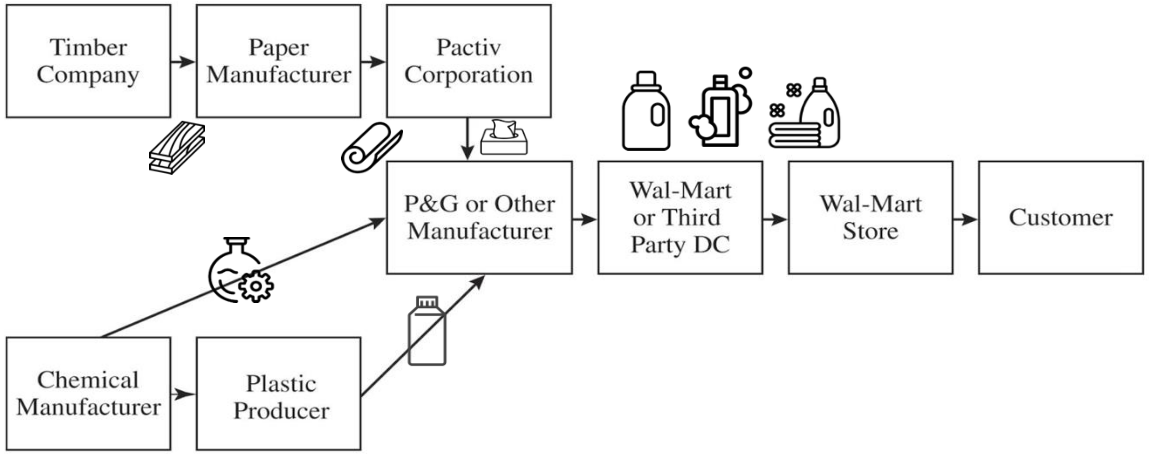
- Limitations of Applying ML to Supply Chains
  - Use of only mechanistic approaches in prior models
    - Hand-engineered input-output relationships of products
    - Fixed links between suppliers and buyers


Machine Learning (ML) Models and Supply Chain (SC) Applications
Demand/Supply forecasting
SC Resilience/risk management
Procurement
SC network optimisation
Food/Retailing supply chains
Energy supply chains
Pharmaceutical supply chains
Logistics/supply chain platforms
Agricultural supply chains
SC Risk prediction
SC risk monitoring
Supply chain resilience
Sustainable/ethical procurement
Public procurement
Resilience in procurement
Factory/warehouse upgrade/repurposing
Last-mile logistics
Inventory management

$$Y(i) = \sum_j A(i,\, j)Y(j) + \overbrace{LFD(i) + E(i) + HD(i) + \sum_j D(j, i)}^{\text{Total Final Demand } (TFD(i))}$$

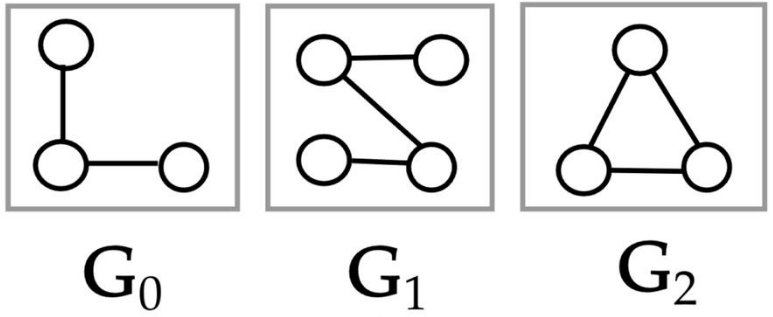
Background

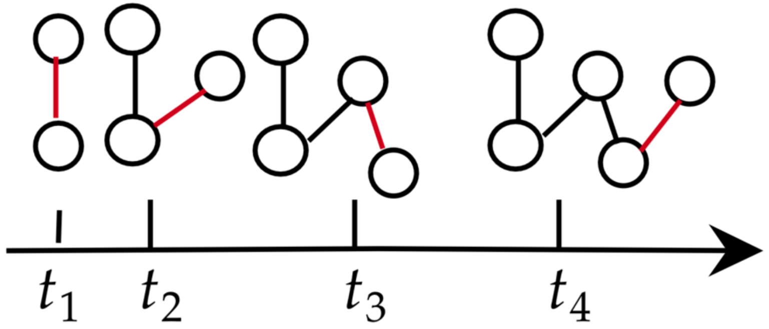
- Supply Chains are Naturally Represented as Graphs
  - Nodes as a firm, edges as transaction between firms
  - A few works about SCs with static GNNs
    - Predicting hidden links between firms, recommending suppliers, and predicting product relations



Problem

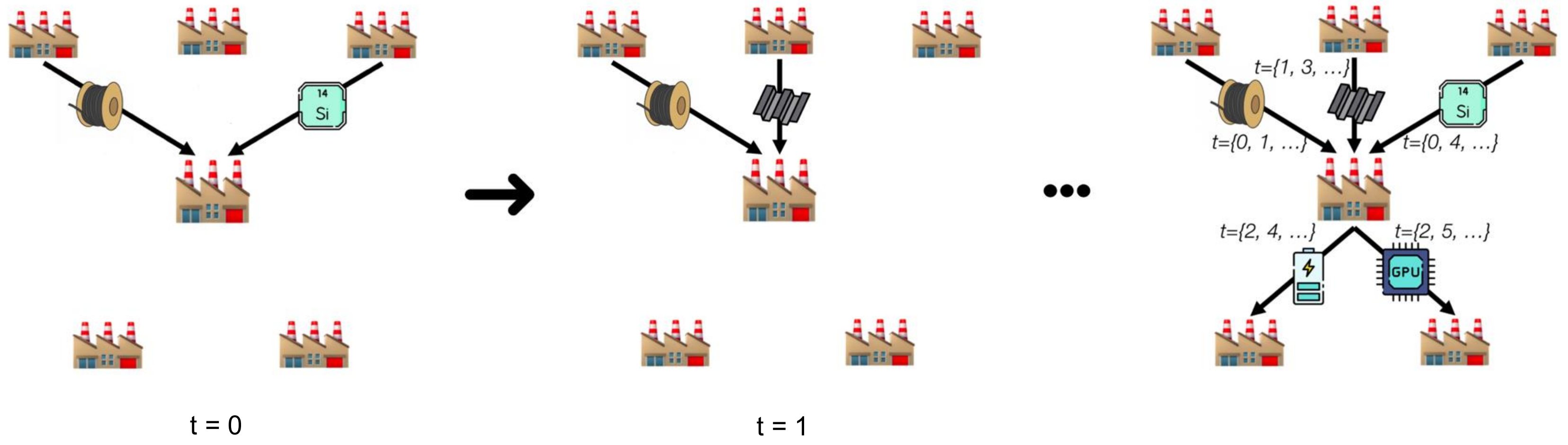
- Limitations of Modeling SCs with GNNs
  - Static GNNs
    - Hard to get crucial insights due to extremely dynamic nature of supply chains
  - Dynamic GNNs
    - Capturing only disruptions propagating across connected firms as transactions
    - Missing the specific connections between each firm's inputs and outputs





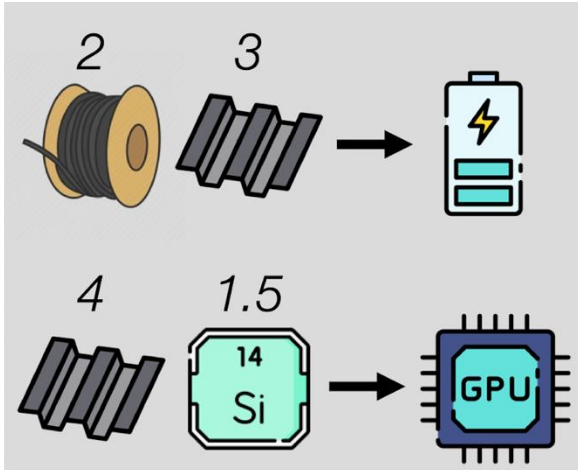
Methodology

- Temporal Production Graphs
  - Directed graphs with time-varying edges and nodes
  - Inputs for products as each node's in-edges
  - Outputs as node's out-edges



Methodology

- Production Functions
  - How much of **input is required to produce an output**
    - How firms internally transform the products they buy into products they supply
  - The structure of **input-output networks among products**
    - Which firms are connected to each other as well as the timing of transactions



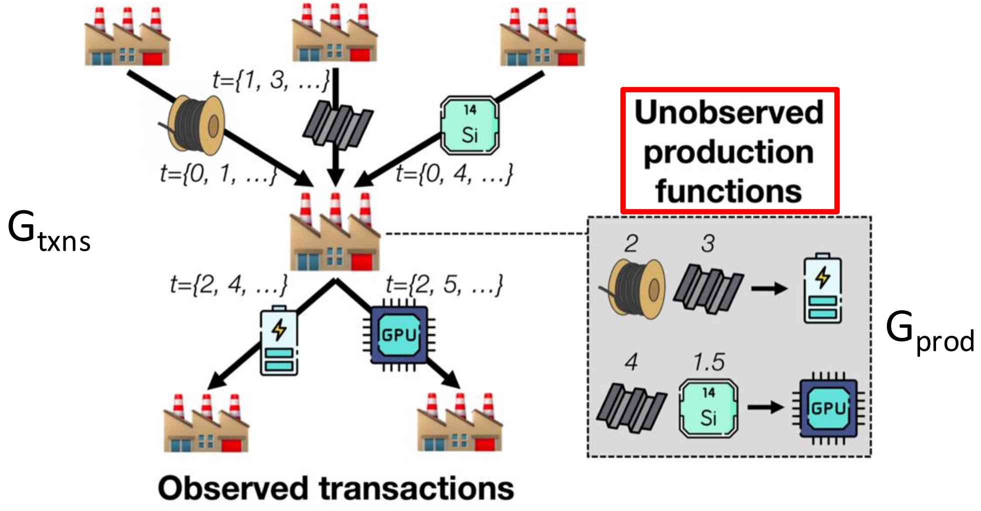
$$\text{Production Function for } p_o = \{\alpha_{p_1 p_o}, \alpha_{p_2 p_o}, \ldots, \alpha_{p_m p_o}\}$$

Car = [4, 1, 2, ...]

$$\alpha_{p_1 p_2} = \mathrm{ReLU}(\mathbf{z}_{p_1} \boxed{\mathbf{W}_{\mathrm{att}}} \mathbf{z}_{p_2} + \boxed{\nu_{p_1 p_2}})$$

Methodology

- Main Goals of TPGs
  - Learning **production functions** of each product
  - Embedding nodes and **predicting future transactions** and **amounts**



Methodology

- Explains of Inventory Module
  - Each firm's **inventory of products**
  - Updated based on bought and consumed products
    - Direct computation of purchase amount per product
    - Lack of visibility into consumed products

$$\underline{\mathbf{x}_i^{(t+1)}} = \max(0, \mathbf{x}_i^{(t)} + \mathbf{b}_i^{(t)} - \mathbf{c}_i^{(t)})$$

$$\text{buy}(i,p,t) = \sum_{e(s,i,p,t)\in\mathcal{E}} \text{amt}(s,i,p,t)$$

$$\text{cons}(i,p,t) = \sum_{e(i,b,p_s,t)\in\mathcal{E}} \boxed{\alpha_{p_s p}} \cdot \text{amt}(i,b,p_s,t)$$

$$\underline{\alpha_{p_1 p_2} = \text{ReLU}(\mathbf{z}_{p_1} \mathbf{W}_{\text{att}} \mathbf{z}_{p_2} + \nu_{p_1 p_2})}$$

Methodology

- Roles of Inventory Module
  - **Penalizing** consumption exceeding inventory
    - Adjust attention weight
  - **Rewarding** high-consumption
    - Without it, $\alpha_{p1,\, p2}$ can be zero

$$\ell_{\text{inv}}(i,t) = \boxed{\lambda_{\text{debt}} \sum_{p\in[m]} \max(0, \text{cons}(i,p,t) - \mathbf{x}_i^{(t)}[p])} \; \boxed{- \lambda_{\text{cons}} \sum_{p\in[m]} \text{cons}(i,p,t)} \Rightarrow \ell_{\text{inv}}(t) = \frac{1}{n}\sum_{i\in[n]} \ell_{\text{inv}}(i,t) + \lambda_{L_2}\sqrt{\sum_{p_1,p_2\in[m]} \nu_{p_1 p_2}^2}.$$

Methodology

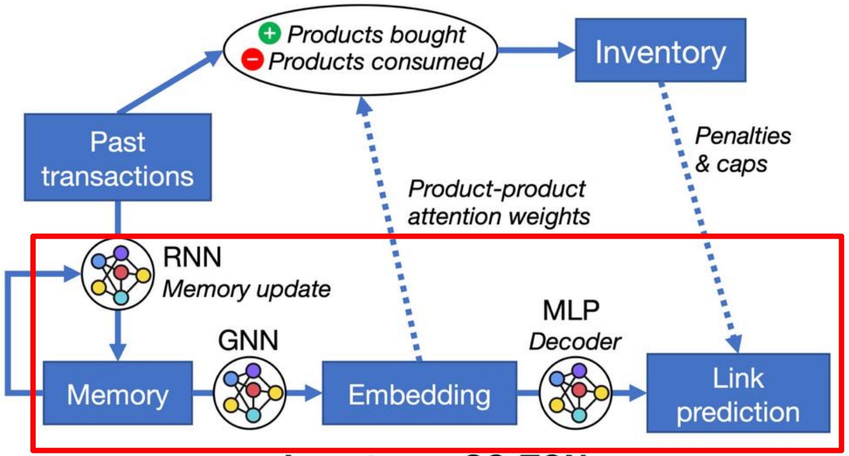
- Roles of Inventory Module
  - Giving **penalties** or **placing caps** for link prediction
    - Penalties to impossible transactions
    - The maximum producible amounts of products as a cap

$$\hat{y}(s,b,p,t) = \mathrm{MLP}([\mathbf{z}_s^{(t)}|\mathbf{z}_b^{(t)}|\mathbf{z}_p^{(t)}]). \quad \boxed{\text{plus}} \quad pen(s,b,p,t) = -\sum_{p' \in [m]} \max(0, \alpha_{pp'} - \mathbf{x}_s^{(t)}[p']).$$

$$\hat{y}_e = \text{predicted amount} \quad \boxed{\underline{\text{min()}}} \quad cap(s,b,p,t) = \min_{p' \in [m]; \alpha_{pp'} > 0} \{\frac{\mathbf{x}_s^{(t)}[p']}{\alpha_{pp'}}\}$$

Methodology

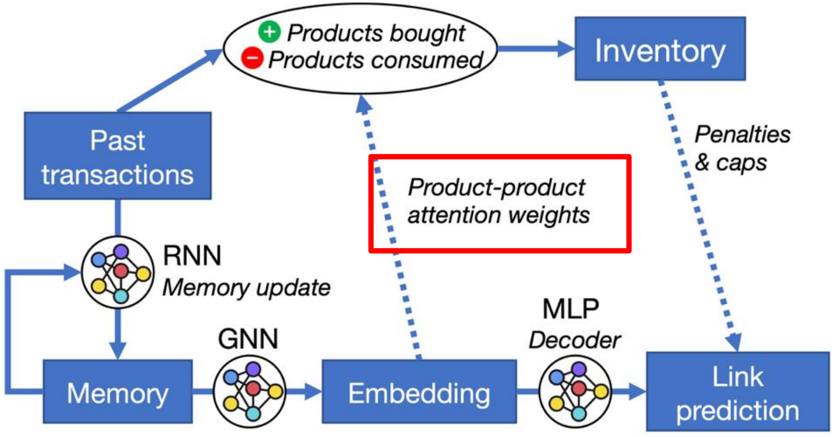
- Overall Processes of TPGs – GNNs
  - Node embedding through TGN
  - Link prediction with **penalties and caps from inventory**



Inventory + SC-TGN

Methodology

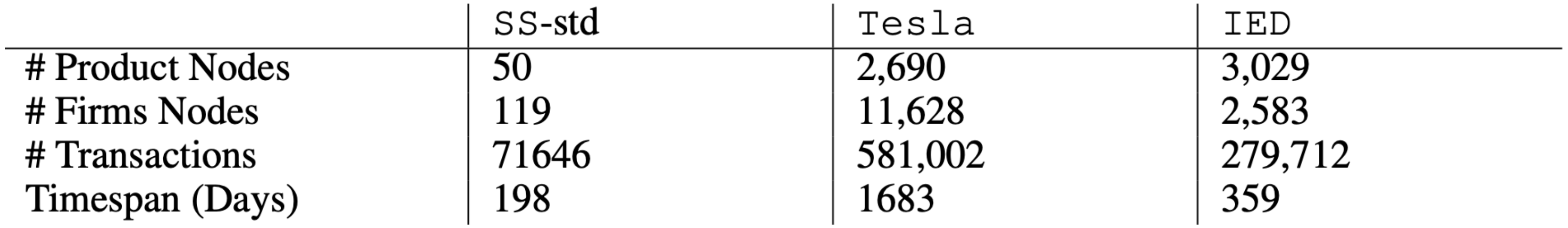
- Overall Processes of TPGs – Production Functions
  - Adjusting a weight matrix based on inventory loss and prediction loss
  - Indirectly **adjusting attention weights and production functions**



Inventory + SC-TGN

Experiment

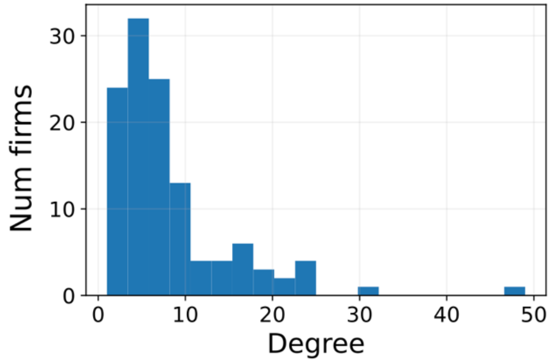
- Dataset Statistics
  - Real-world data
    - Tesla (1/1, 2019 to 12/31, 2022), IED (2023)
  - Synthetic data
    - SupplySim (SS)

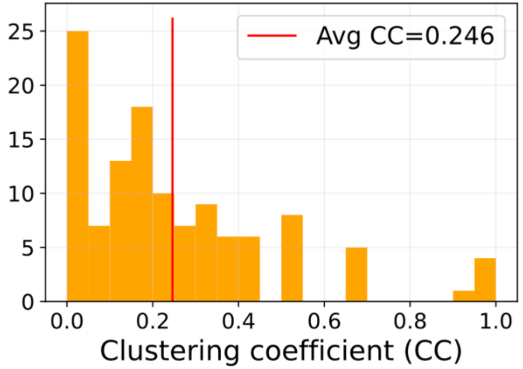


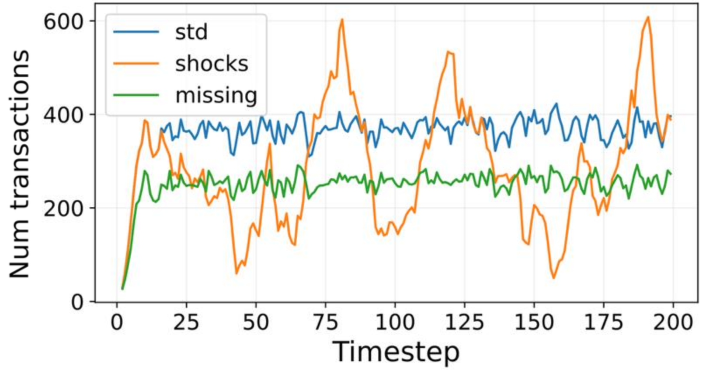
	SS-std	Tesla	IED
# Product Nodes	50	2,690	3,029
# Firms Nodes	119	11,628	2,583
# Transactions	71646	581,002	279,712
Timespan (Days)	198	1683	359

SupplySim

- Why Do Experiments Use Synthetic Data?
  - Absence of production functions
    - Lack of the ground-truth
  - Real-world data derived from bills of lading
    - Contains only international transactions, excluding domestic ones

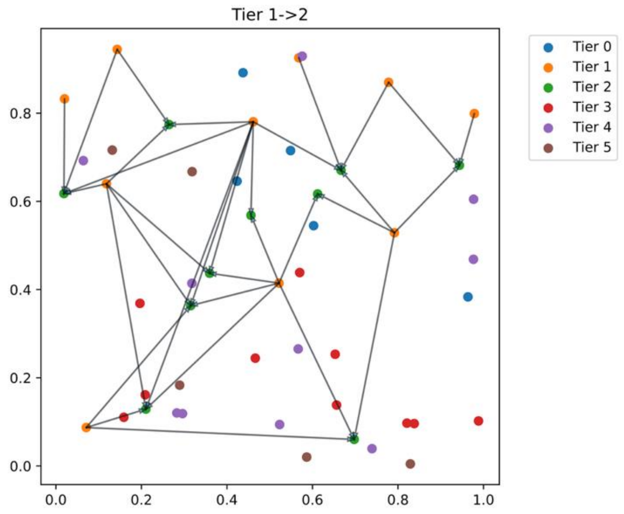






Constructing Graph with SupplySim

- Constructing the Production Graph, G_{prod}
  - Classify products into tiers, 0 to 5
  - Connect each product to 2 - 4 nearest products in the previous tier
    - Ensuring shared components with similar products

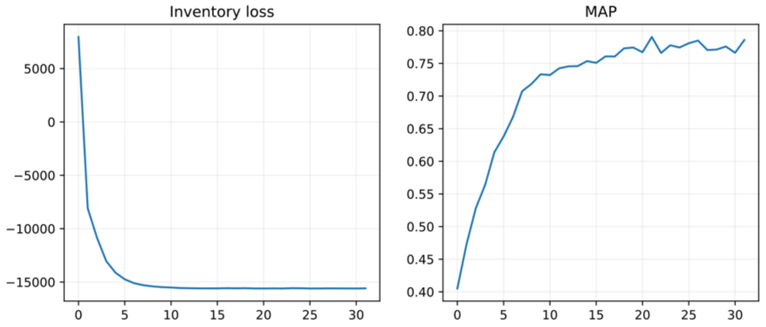


Constructing Graph with SupplySim

- Constructing Supplier-Buyer Graph
  - Link of each embedded product to firms capable of producing it
  - Generate of candidate firms from G_{prod} for each product and assignment of suppliers
- Generating Transactions
  - Apply the agent-based ARIO model to generate probabilistic transaction data

Experiment

- Results for Production Function Learning
  - Evaluated with mean average precision



	SS-std	SS-shocks	SS-missing	IED
Random baseline	0.124 (0.009)	0.124 (0.009)	0.124 (0.009)	0.060 (0.002)
Temporal correlations	0.745	0.653	0.706	0.128
PMI	0.602	0.602	0.606	0.175
node2vec	0.280	0.280	0.287	0.127
Inventory module (direct)	0.771 (0.005)	0.770 (0.006)	0.744 (0.006)	0.143 (0.004)
Inventory module (emb)	**0.790** (0.005)	**0.778** (0.011)	**0.755** (0.007)	**0.262** (0.005)

Experiment

- Results for Predicting Existence of Future Edges
  - Evaluated with mean reciprocal rank

$$r_{\text{opt}}(e) = \sum_{n \in \mathcal{N}_e} \mathbb{1}[\hat{y}_n < \hat{y}_e]$$

$$r_{\text{pes}}(e) = \sum_{n \in \mathcal{N}_e} \mathbb{1}[\hat{y}_n \leq \hat{y}_e].$$

$$\text{MRR} = \frac{1}{|B|} \sum_{e \in B} \left(\frac{r_{\text{opt}}(e) + r_{\text{pos}}(e)}{2} + 1 \right)^{-1}.$$

	SS-std	SS-shocks	SS-missing	Tesla	IED
Edgebank (binary)	0.174	0.173	0.175	0.131	0.164
Edgebank (count)	0.441	0.415	0.445	0.189	0.335
Static	0.439 (0.001)	0.392 (0.002)	0.442 (0.001)	0.321 (0.001)	0.358 (0.001)
Graph transformer	0.431 (0.003)	0.396 (0.024)	0.428 (0.003)	0.507 (0.020)	0.613 (0.045)
SC-TGN	0.522 (0.003)	0.449 (0.004)	**0.494** (0.004)	**0.820 (0.007)**	**0.842 (0.004)**
SC-TGN+inv	**0.540** (0.003)	**0.461** (0.009)	0.494 (0.004)	0.818 (0.004)	0.841 (0.008)
SC-GraphMixer	0.453 (0.005)	0.426 (0.004)	0.446 (0.003)	0.690 (0.027)	0.791 (0.009)
SC-GraphMixer+inv	0.497 (0.004)	0.448 (0.004)	0.446 (0.002)	0.681 (0.014)	0.791 (0.008)

Experiment

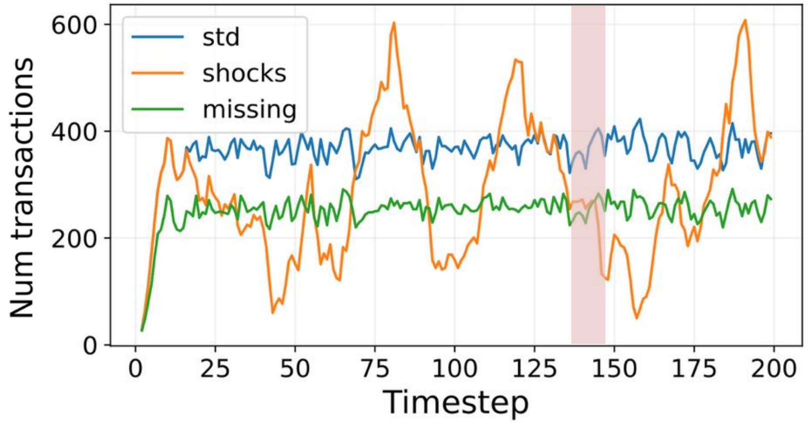
- Results for Predicting Weight of Future Edges
  - Evaluated with root mean squared error

$$RMSE = \sqrt{\frac{1}{|B|}\sum_{e \in B}(amt(e) - \hat{y}_e)^2}.$$

	SS-std	SS-shocks	SS-missing	Tesla	IED
Edgebank (avg)	0.341	0.387	0.349	1.148	0.489
Static	0.343 (0.008)	0.425 (0.019)	0.374 (0.027)	1.011 (0.007)	0.504 (0.018)
Graph transformer	0.340 (0.005)	0.398 (0.025)	0.361 (0.016)	0.885 (0.024)	0.425 (0.008)
SC-TGN	**0.303** (0.003)	**0.359** (0.007)	0.313 (0.002)	0.796 (0.012)	0.428 (0.011)
SC-TGN+inv	0.312 (0.003)	0.370 (0.009)	**0.312** (0.002)	0.801 (0.015)	**0.422 (0.011)**
SC-GraphMixer	0.318 (0.003)	0.384 (0.005)	0.330 (0.005)	0.774 (0.077)	0.457 (0.008)
SC-GraphMixer+inv	0.320 (0.004)	0.378 (0.005)	0.328 (0.003)	**0.767 (0.054)**	0.454 (0.012)

Experiment

- Generating Future Transactions under Shocks
  - Evaluated with AUROC on transaction existence scores
  - Robust under more challenging settings, where sudden shocks occur



t	AUROC (all)	AUROC (train)
139	0.996	0.824
140	0.986	0.733
141	0.966	0.721
142	0.938	0.637
143	0.943	0.637
144	0.936	0.593
145	0.936	0.634

Conclusion

- ☐ TPGs represent supply chains as temporal graphs with production functions
- ☐ The inventory module is used to learn invisible production functions
- ☐ Transactions of firms can be inferred automatically through production functions
- ☐ TPGs demonstrate robustness in various scenarios using both real and synthetic data

Thank you