Scalable and Efficient Temporal Graph Representation Learning via Forward Recent Sampling

Yuhong Luo
Rutgers University

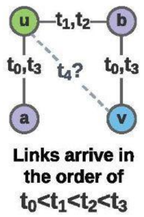
Pan Li
Georgia Institute of Technology

LOG CONFERENCE

Overview

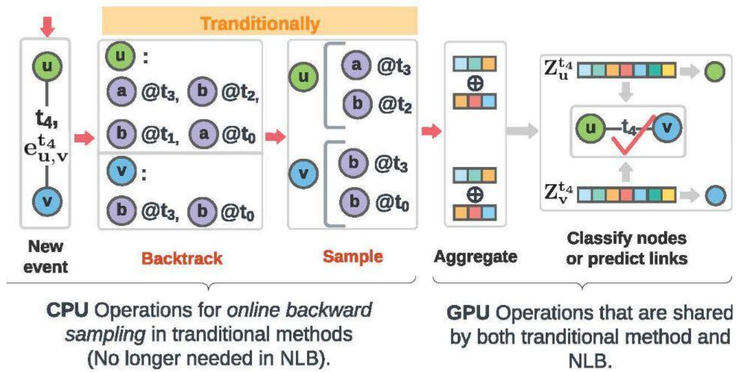
TL; DR: We propose a scalable framework that employs a novel "forward recent sampling" strategy to aggregate historical node interactions

Background



Problem: Learn node representations
Downstream tasks:
1. Predict future *links* 2. Classify *nodes*
Useful Information:
1. *Recent* historical interactions
2. Self and neighbors' current statuses

Traditionally: Existing methods for TGRL rely on **backtracking and sampling** neighboring nodes through the interaction history of each node.



S1: Truncating most recent neighbors, no sampling
S2: Uniform Sampling across the entire history
S3: Recent Sampling with prob. $\exp(-c\Delta t)$, $c > 0$

Computationally inefficient because:
- Need to look back for historical interactions
- Need to compute the sampled neighbors online
- Cannot parallelize for a batch of nodes in GPUs

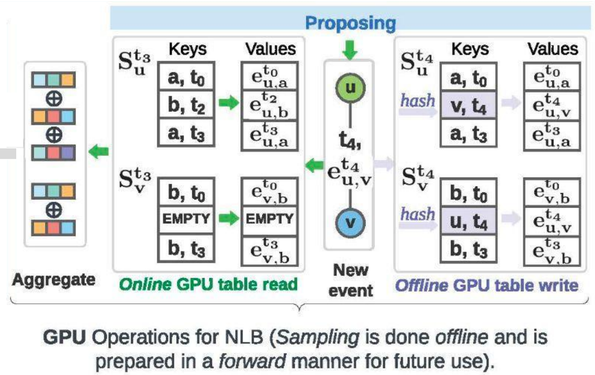
Forward Recent Sampling

We present **N**o-**L**ooking-**B**ack (**NLB**) that adopts forward recent sampling, which
1. Avoids online neighbor sampling.
2. Achieves the sampling for node batches in GPUs
3. Learns predictive node representations for both *edge* and *node* level tasks.

Insight: Maintain the downsampled neighbors for each node within a GPU hash table.

Hash Keys: Neighbor IDs and interacting time
Hash Values: Edge features

Inference:
- Directly access the hash tables for the downsampled neighbors of the targeted nodes in *parallel*.
- Aggregate the sampled neighbors' statuses via *Attn*.
- Generate node representations for predictions.



Maintenance:
- For new link *(u,v,t)*, initialize update to tables S_u & S_v.
- Locate indices of the hash table with keys *(v,t)* & *(u,t)*.
- If hash collision occurs, replace entry with prob. α.
- Otherwise, insert new keys and values.

Computation benefits
1. No online sampling 2. Parallel processing in GPUs

Theoretical and Empirical Results

Theoretically: Matches recent sampling assuming links come in for any node via a *Poisson point process* with constant intensity λ.

Empirically:
1. Significantly improves inference latency.
2. Reduces training time and energy consumption.
3. Achieve comparable prediction performances.

Measurement	GDELT	MAG	Ubuntu	Wiki-talk
nodes	17K	122M	159K	1.1M
temporal links	191M	1.3B	964K	7.8M
node classes	81	152	0	0

Including million-to-billion-scale datasets

Method		Train	Test	Inf. Lat.	CPU (MJ)	GPU (MJ)		Train	Test	Inf. Lat.	CPU (MJ)	GPU (MJ)
TGN-trunc	Ubuntu	15.95	2.05	1.37	3,332	1.75	GDELT	1977	406	302	431,567	174.6
TGN-unif		15.55	2.11	1.42	3,326	1.78		4791	1013	777	797,311	389.4
TGAT-trunc		12.37	1.70	1.20	3,028	1.30		1123	248	183	306,860	92.7
TGAT-unif		13.64	1.93	1.27	3,254	1.37		1493	383	301	251,452	83.8
APAN-trunc		12.96	1.94	0.61	2,802	1.36		1530	338	200	485,818	211.2
APAN-unif		13.01	2.07	0.62	2,894	1.34		1839	347	187	512,061	188.4
NAT		17.52	1.96	1.22	2,853	1.27		-	-	-	-	-
NAT-node		15.60	1.60	1.30	2,824	1.23		-	-	-	-	-
NLB-edge		11.06	1.30	0.57	2,764	2.24		1323	139	63	252,267	307.6
NLB-node		10.74	1.35	0.59	2,748	2.17		1089	118	60	215,820	258.5
Method		Train	Test	Inf. Lat.	CPU (MJ)	GPU (MJ)		Train	Test	Inf. Lat.	CPU (GJ)	GPU (GJ)
TGN-trunc	Wiki-talk	74.63	11.76	7.61	16,715	9.57	MAG	6845	1266	795	15,041	1.59
TGN-unif		85.96	13.77	8.19	18,925	11.2		5704	1224	741	21,334	2.02
TGAT-trunc		58.37	8.74	5.88	14,787	5.90		3465	754	518	10,427	1.06
TGAT-unif		63.32	9.14	6.35	16,553	6.75		3520	774	541	13,236	1.35
APAN-trunc		60.68	10.34	2.98	13,898	6.94		12347	1600	809	22,262	2.45
APAN-unif		64.06	10.61	2.88	14,463	7.04		-	-	-	-	-
NAT		81.32	10.03	6.08	14,736	7.13		-	-	-	-	-
NAT-node		67.28	8.79	4.88	14,492	6.82		-	-	-	-	-
NLB-edge		59.18	7.30	3.03	12,277	15.0		3073	593	296	2,684	1.73
NLB-node		56.36	7.01	2.99	11,519	13.6		2729	577	279	2,870	1.69

Table 4: Scalability evaluation on all datasets. Note that MJ = 10^6 Joules, and GJ = 10^9 Joules. Note that TGL is adopted to implement baselines including TGN, TGAT, and APAN.

1.32-4.40× **faster** training, 1.2-7.94× more **energy efficient**, and 1.63-12.95× **lower inference latency** v.s. baselines.

Matches or outperforms SOTA on *link* and *node* tasks. Benefits from both *recency* and *randomness*.

Method	Wikipedia (AUC)	Reddit (AUC)	GDELT (F1)	MAG (F1)
TGN-trunc	86.85 ± 0.56	63.45 ± 1.21	18.39 ± 1.37	2.33 ± 0.00
TGN-unif	85.99 ± 0.62	**64.38 ± 1.60**	21.38 ± 0.04	2.33 ± 0.00
TGAT-trunc	78.27 ± 1.12	60.54 ± 0.97	22.91 ± 0.03	2.33 ± 0.00
TGAT-unif	84.08 ± 0.41	63.44 ± 2.72	**24.34 ± 0.02**	2.33 ± 0.00
APAN-trunc	86.50 ± 0.60	56.74 ± 0.62	9.14 ± 0.43	3.11 ± 0.00
APAN-unif	87.27 ± 1.02	60.68 ± 1.42	9.90 ± 0.01	CPU OOM
NAT	-	-	-	-
NAT-node	86.65 ± 0.55	60.44 ± 2.36	GPU OOM	GPU OOM
NLB-edge	**89.92 ± 0.20**	62.73 ± 2.27	23.90 ± 0.05	**30.93 ± 0.02**
NLB-node	89.52 ± 0.06	62.93 ± 0.56	23.46 ± 0.08	29.69 ± 0.02

Table 3: Performance of **node classification**.

Task	Method	Wikipedia	Reddit	GDELT	MAG	Ubuntu	Wiki-talk
Transductive	TGN-trunc	99.48 ± 0.07	99.65 ± 0.07	98.63 ± 0.05	99.32 ± 0.06	76.09 ± 0.17	84.73 ± 3.57
	TGN-unif	99.44 ± 0.03	99.66 ± 0.05	97.85 ± 0.02	99.24 ± 0.03	78.99 ± 1.69	83.61 ± 0.10
	TGAT-trunc	97.47 ± 0.06	97.84 ± 0.03	98.31 ± 0.02	99.09 ± 0.03	76.71 ± 0.33	81.23 ± 0.06
	TGAT-unif	95.35 ± 0.18	98.15 ± 0.15	97.76 ± 0.01	99.02 ± 0.05	78.59 ± 0.23	80.48 ± 0.01
	APAN-trunc	99.20 ± 0.03	96.46 ± 2.98	97.89 ± 0.16	90.61 ± 1.24	69.90 ± 3.28	82.10 ± 5.67
	APAN-unif	97.90 ± 0.40	97.65 ± 0.20	97.56 ± 0.74	CPU OOM	77.62 ± 2.71	84.32 ± 7.28
	NAT	**99.72 ± 0.03**	**99.90 ± 0.01**	GPU OOM	GPU OOM	**89.65 ± 0.17**	**94.66 ± 0.03**
	NAT-node	99.29 ± 0.07	99.76 ± 0.02	GPU OOM	GPU OOM	87.48 ± 0.49	93.03 ± 0.52
	NLB-edge	99.42 ± 0.08	99.73 ± 0.02	**98.94 ± 0.28**	**99.39 ± 0.02**	87.18 ± 0.55	91.72 ± 0.62
	NLB-node	99.03 ± 0.07	99.67 ± 0.02	**98.80 ± 0.16**	99.15 ± 0.02	87.48 ± 0.64	91.95 ± 0.17
Inductive	TGN-trunc	98.55 ± 0.08	99.44 ± 0.06	98.62 ± 0.01	96.05 ± 0.12	80.70 ± 1.64	87.65 ± 1.01
	TGN-unif	98.33 ± 0.08	99.30 ± 0.01	**98.77 ± 0.08**	96.45 ± 0.26	**85.23 ± 1.66**	87.11 ± 0.30
	TGAT-trunc	96.57 ± 0.04	95.22 ± 0.21	94.36 ± 0.01	98.80 ± 0.01	77.22 ± 0.07	79.97 ± 0.04
	TGAT-unif	93.80 ± 0.15	96.12 ± 0.07	94.05 ± 0.02	98.77 ± 0.01	78.07 ± 0.04	77.03 ± 0.03
	APAN-trunc	96.65 ± 0.07	95.93 ± 2.43	98.48 ± 0.01	CPU OOM	65.19 ± 5.50	78.90 ± 1.11
	APAN-unif	97.23 ± 0.31	96.56 ± 0.32	97.64 ± 0.14	CPU OOM	78.39 ± 2.87	87.86 ± 0.11
	NAT	**99.52 ± 0.03**	**99.78 ± 0.03**	GPU OOM	GPU OOM	84.71 ± 1.01	**92.23 ± 1.16**
	NAT-node	98.47 ± 0.25	99.53 ± 0.21	GPU OOM	GPU OOM	81.51 ± 1.36	79.12 ± 5.14
	NLB-edge	98.43 ± 0.26	99.43 ± 0.07	98.16 ± 0.32	**98.85 ± 0.02**	**86.88 ± 0.62**	90.91 ± 1.14
	NLB-node	98.31 ± 0.30	99.36 ± 0.09	97.14 ± 0.42	**98.79 ± 0.11**	85.47 ± 0.73	**91.22 ± 1.39**

Table 2: **Link prediction** performance in AUC.