Scaling Attributed Network Embedding to Massive Graphs

by: R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. Bhowmick

Basic data analytics is easy.

Attributes

	X_1	X_2	X_3	X_4	$Y?$
Stock	**Profit**	**Revenue**	**Market share**	**Overvalued?**	**Buy?**
TSLA	$721m	$31B	3.6%	Yes	???

Target attribute

Graph data analytics is more powerful.

X1	X2	X3	X4	X5
$$$	$$$	$$$	$$$	$$$

X1	X2	X3	X4	X5
$$$	$$$	$$$	$$$	$$$

X1	X2	X3	X4	X5
$$$	$$$	$$$	$$$	$$$

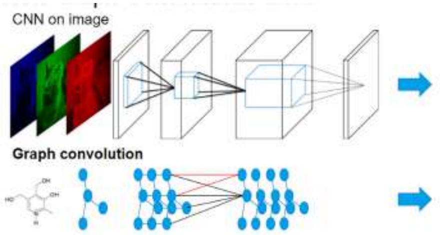
X1	X2	X3	X4	X5
$$$	$$$	$$$	$$$	$$$

Stock	Profit	Revenue	Market share	Overvalued?	Buy?
TSLA	$721m	$31B	3.6%	Yes	???

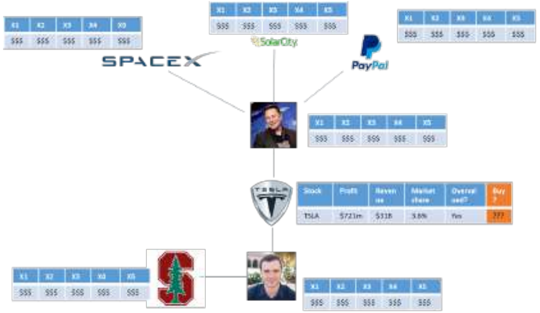
X1	X2	X3	X4	X5
$$$	$$$	$$$	$$$	$$$

X1	X2	X3	X4	X5
$$$	$$$	$$$	$$$	$$$

Graph data analytics is powerful but difficult.

	Single table data analytics	Attributed graph data analytics
Tools	dmlc XGBoost + scikit learn	 Deep graph neural network
Difficulty	★ ★	★ ★ ★ ★ ★
Req. Skill level		

We present Practical Attributed Network Embedding (PANE).



PANE

Embedding vector

X1	X2	X3	X4	...	X30	X31	X32	Buy?
0.12	0.34	0.56	0.78	...	0.101	0.223	0.33 4	???

Attributed graph data analytics

Embedding vector data analytics

Performance of PANE

<ins>*Effective*</ins>

Accuracy (F1):

up to +17.2% ⬆

Compared to SOTA
Neural Network methods

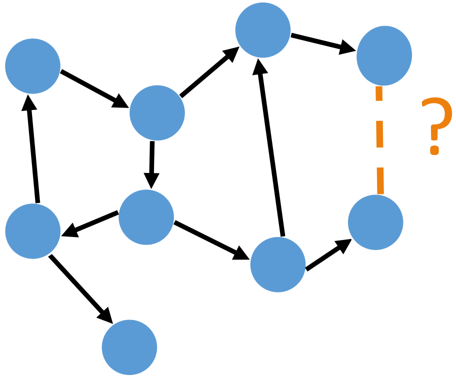
Computing all embeddings on a HUGE graph with 59m nodes, 0.98b edges, 2k attributes

Single-server:

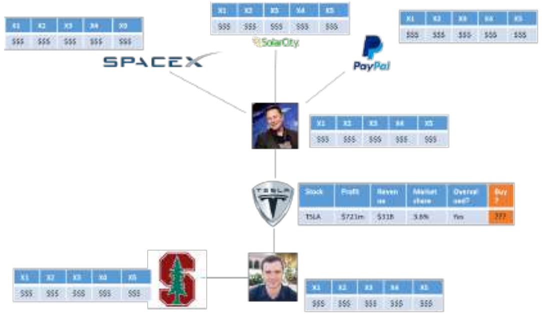
~ 12 hours

<ins>*Efficient*</ins>

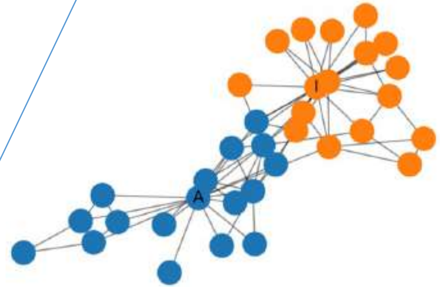
Applications of PANE



Link Prediction



Attribute Inference



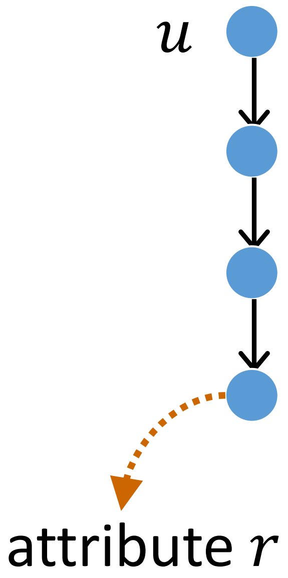
Node Classification

PANE is based on mostly novel database technologies (with a bit of machine learning flavor).

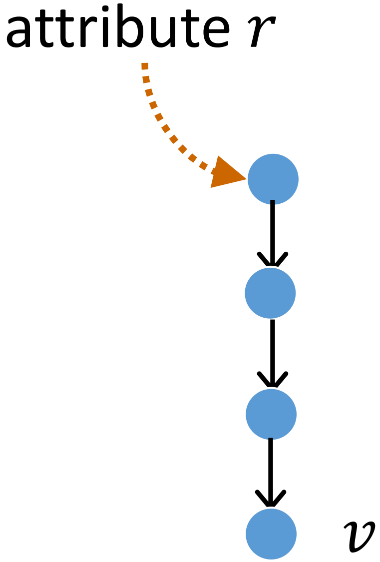
1. PANE measures Node-Attribute affinity via random walks.
2. PANE computes embeddings with joint matrix factorization.
3. PANE makes full use of multi-core parallel computation.

Types of Random Walks in PANE

1



Forward: node-to-attribute



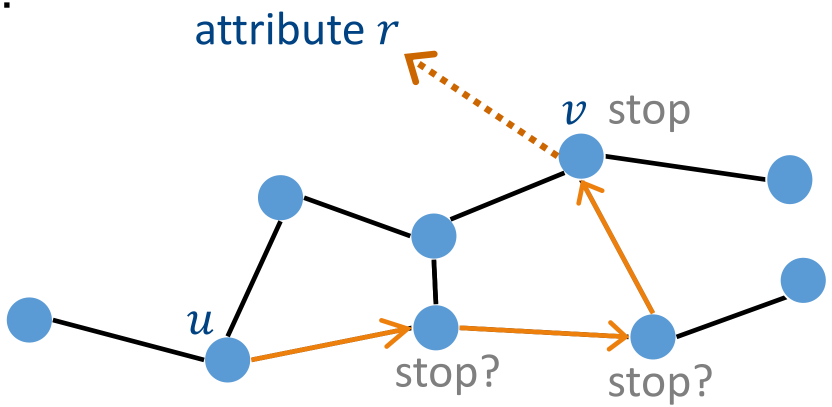
Backward: attribute-to-node

Forward Random Walks

1

- Forward random walk from node u:
  - Start from u
  - At each step, stop with α probability
  - After stopping at a node v, pick an attribute r with probability $\propto \mathrm{w}(v, r)$

- Intuition: it samples an attribute r from the vicinity of u

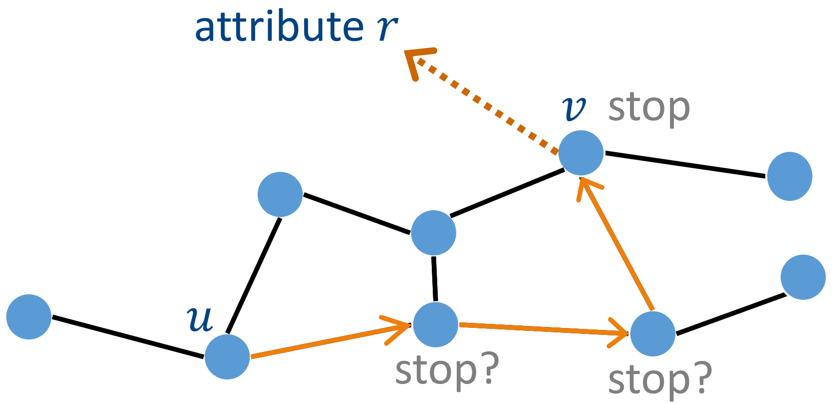


Node-Attribute Affinity

1

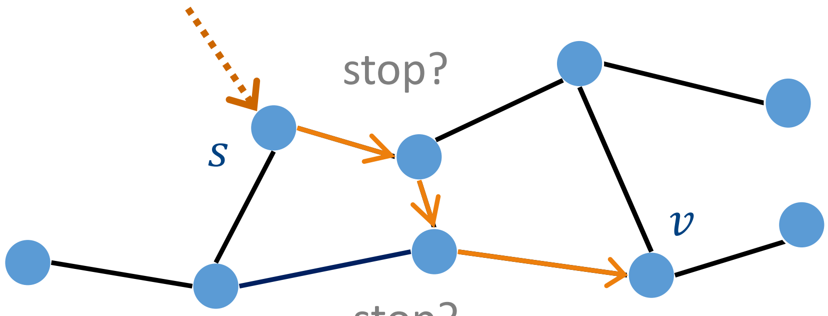
<u>Node-attribute affinity:</u>

$\mathbf{F}[u, r] =$ normalized probability that a forward random walk from u samples r in the end



Backward RW & Attribute-Node Affinity

- Backward random walk from attribute r
  - Randomly pick a node s with probability proportional to the weight of (s, r)
  - Start a random walk from s
  - At each step, stop with α probability
  - Let v be the stopping point of the walk



<u>Attribute-node affinity</u>

$\mathbf{B}[r, v] \leftarrow$ normalized random walk probability from attribute r to node v

Node-to-Node affinity is derived.

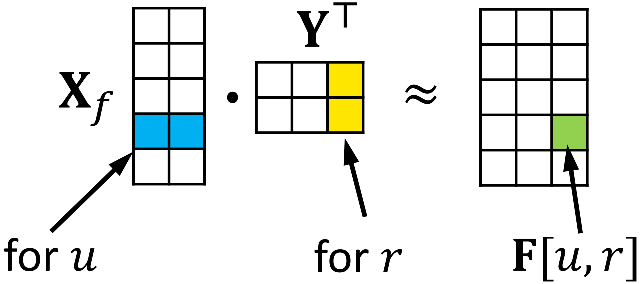
1

$$p(u,v) = \sum_{r \in R} \mathbf{F}[u,r] \cdot \mathbf{B}[r,v]$$

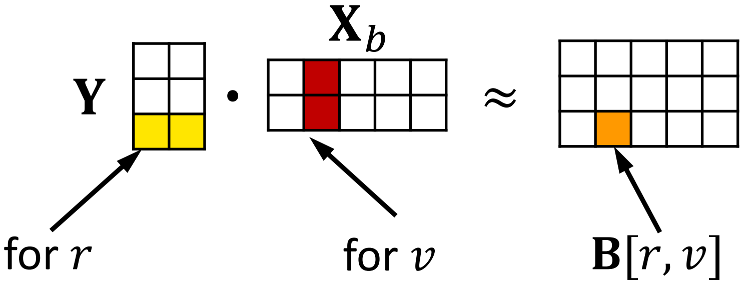
This saves a LOT of space: $O(n^2) \rightarrow O(nd), d \ll n$

Embedding Matrices in PANE

- We construct
  - two embedding matrices $\mathbf{X}_f$ and $\mathbf{X}_b$ for the nodes, and
  - one embedding matrix $\mathbf{Y}$ for attributes



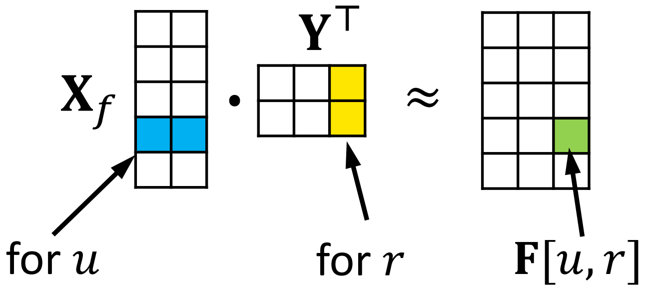
- Optimization objective:
  - $\mathbf{X}_f \cdot \mathbf{Y}^\top \approx \mathbf{F}$, to capture node-attribute affinity
  - $\mathbf{Y} \cdot \mathbf{X}_b^\top \approx \mathbf{B}$, to capture attribute-node affinity



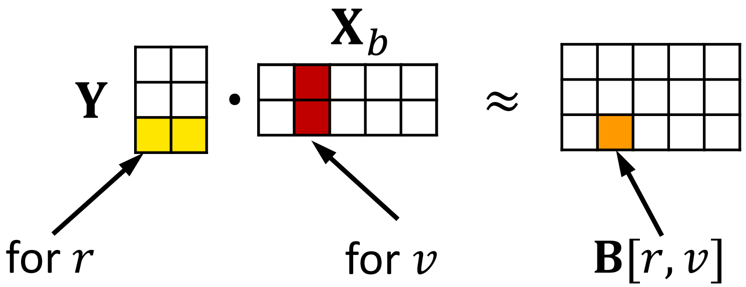
Solving the optimization program

2

- Jointly factorize $\mathbf{F}$ and $\mathbf{B}$ to obtain $\mathbf{X}_f$, $\mathbf{X}_b$, and $\mathbf{Y}$
  - Formulate the joint factorization as a least square problem
  - Solve it using gradient descent
  - Use randomized SVD to obtain a good initial solution



- Time complexity: $O(mdt + ndkt)$
  - k is the embedding size
  - t is the number of iterations ($t = 5$ in our experiments)



Greedy Initialization + SGD

2

$$\mathbf{F} \approx \boxed{\mathbf{U} \cdot \mathbf{\Sigma}} \cdot \boxed{\mathbf{V}^{\mathrm{T}}}$$

- $\mathbf{X}_f = \mathbf{U} \cdot \mathbf{\Sigma}, \mathbf{Y} = \mathbf{V}$
- $\mathbf{V}$=$\mathbf{Y}$ is unitary
- $\mathbf{Y}^{\mathrm{T}} \cdot \mathbf{Y} = \mathbf{I}$
- $\mathbf{X}_b = \boxed{\mathbf{X}_b \mathbf{Y}^{\mathrm{T}}} \mathbf{Y} = \mathbf{B} \cdot \mathbf{Y}$

Greedy initialization of embeddings via randomized SVD and the unitary property

For t iterations:

Update $\mathbf{X}_f, \mathbf{X}_b$ via SGD;

Update $\mathbf{Y}$ via SGD;

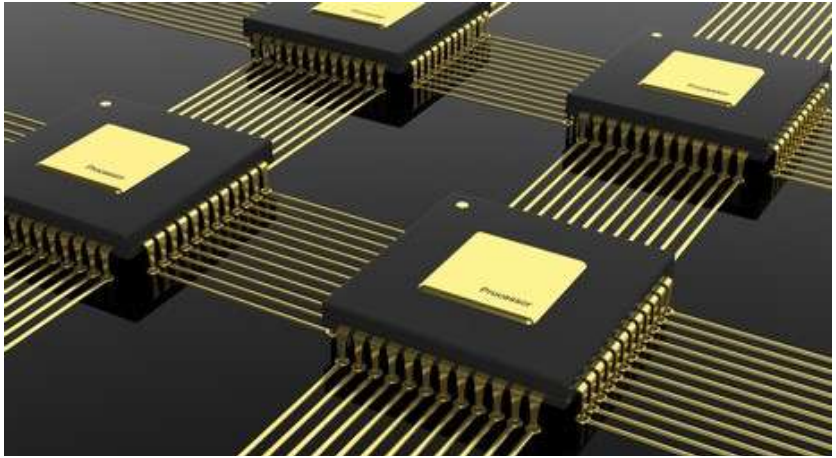
$$\mathbf{F} \approx \mathbf{X}_f \cdot \mathbf{Y}$$

$$\mathbf{B} \approx \mathbf{X}_b \cdot \mathbf{Y}$$

Only a few iterations are needed!

PANE is fully parallelized on multi-core computers.

3



Explained in Section 4 of our paper.

Experiments: 8 Datasets

Name	# of nodes	# of edges	# of distinct attributes	# of attributes per node	# of distinct labels
Cora	2.7k	5.4k	1.4k	18.2	7
Citeseer	3.3k	4.7k	3.7k	31.9	6
Facebook	4k	88.2k	1.3k	8.3	193
Pubmed	19.7k	44.3k	0.5k	50.2	3
Flickr	7.6k	479.5k	12.1k	24.0	9
Google+	107.6k	13.7M	15.9k	2793.7	468
TWeibo	2.3M	50.7M	1.7k	7.3	8
MAG	59.3M	978.2M	2.0k	7.3	100

Experiments: 10 Competitors

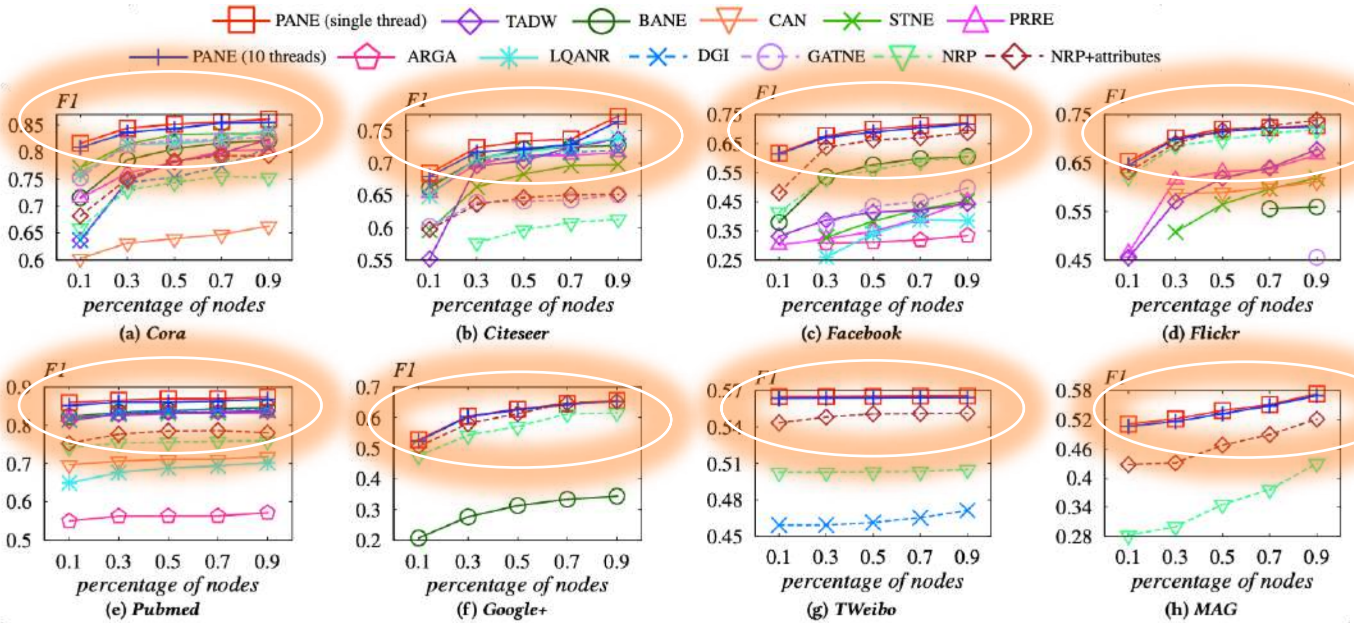
- Default embedding dimensionality: $k = 128$

- 6 neural-network-based methods
  - STNE [KDD 2018]
  - ARGA [IJCAI 2018]
  - LQANR [IJCAI 2019]
  - CAN [WSDM 2019]
  - DGI [ICLR 2019]
  - GATNE [KDD 2019]

- 3 factorization-based methods
  - TADW [IJCAI 2015]
  - BANE [ICDM 2018]
  - NRP [VLDB 2020]

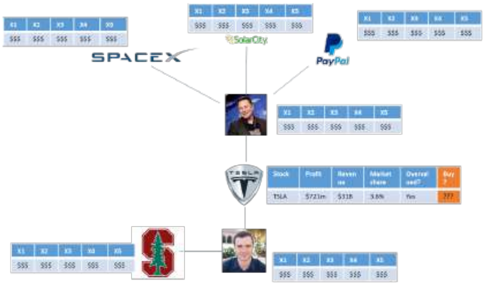
- 1 other method
  - PRRE [CIKM 2018]

Results: Node Classification



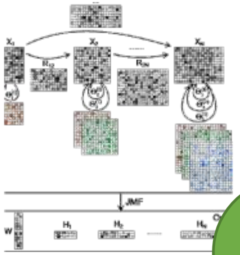
- Percentage of nodes used for training: 10% ~ 90%
- PANE vs. SOTA: improvements of 3.4%-17.2% in terms of F1 measure

THANK YOU



1

Random walks



2

Joint matrix factorization

3

Parallelization


Q
A

Scaling Attributed Network Embedding to Massive Graphs

by: R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. Bhowmick

Code: https://github.com/AnryYang/PANE