Scaling Attributed Network Embedding to Massive Graphs

National University of Singapore

Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, Juncheng Liu and Sourav S. Bhowmick

Contact: renchi@nus.edu.sg

Problems and Motivations

Attributed Graph

- Let $G = (V, E_V, R, \mathbf{P}, \mathbf{R})$ be an attributed network, consisting of
  (i) a node set V with cardinality n,
  (ii) a set of edges E_V of size m, a random walk matrix $\mathbf{P}$
  (iii) a set of attributes R with cardinality d, and
  (iv) a node-attribute matrix $\mathbf{R}$, where $\mathbf{R}[v_i, r_j]$ signifies the strength of the association between node v_i and attribute r_j

Attributed Network Embedding (ANE):

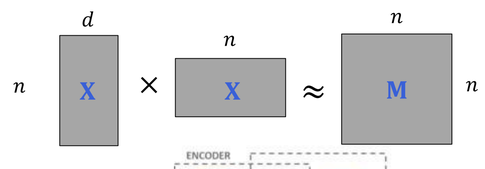
- Given an attributed network G and a space budget $k \ll n$
- Attributed Network Embedding is to compute a length-k embedding $\mathbf{X}_v$ for each node v in the graph, such that $\mathbf{X}_v$ captures the graph structure and attribute information surrounding node v.

Applications

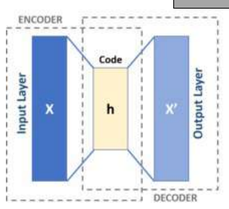
- Node Classification [Hamilton et al. NeurIPS'17, Velickovic et al. ICLR'19]
- Link Prediction and Recommendations: Pinterest's "PinSage" [Yang et al. KDD'18], Alibaba's "AliGraph" [Zhu et al. VLDB'19, Chen et al. KDD'19]
- Attribute Inference [Meng et al. WSDM'18]

Existing Work

- Matrix factorization-based methods
  - Construct a $n \times n$ node-node affinity matrix based on proximity & attribute similarity between nodes, and then factorize the matrix

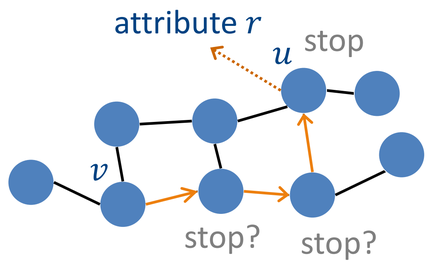


- Neural network-based methods
  - Feed the graph matrix & attribute similarity matrix to a deep neural network to compress them
  - Reconstruct the graph matrix & attribute similarity matrix based on the compress data



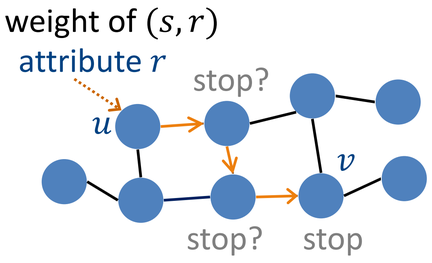
Objective Function

Node-Attribute Affinity (Forward Affinity)



- Start a random walk from u
- At each step, stop with α probability
- After stopping at a node v, pick an attribute r with probability $\propto \mathrm{w}(v, r)$
- $\mathbf{F}[v, r] = \log\left(\frac{n \cdot p_f(v,r)}{\sum_{u \in V} p_f(u,r)} + 1\right)$
- $p_f(u, r)$ is the probability that a node-to-attribute random walk from u samples r in the end
- We normalize $p_f(u, r)$ by how "frequently" r is sampled by others

Attribute-Node Affinity (Backward Affinity)



- Randomly pick a node s with probability $\propto$ 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
- $\mathbf{B}[v, r] = \log\left(\frac{d \cdot p_b(v,r)}{\sum_{r_h \in V} p_b(v,r_h)} + 1\right)$
- $p_b(r, v)$ is the probability that an attribute-to-node random walk from r samples v in the end

Objective Function

$$\min_{\mathbf{X}_f, \mathbf{Y}, \mathbf{X}_b} \sum_{v \in V} \sum_{r \in R} (\mathbf{F}[v, r] - \mathbf{X}_f[v] \cdot \mathbf{Y}[r])^2 + (\mathbf{B}[v, r] - \mathbf{X}_b[v] \cdot \mathbf{Y}[r])^2$$

- $\mathbf{X}_f[v]$: forward node embedding
- $\mathbf{X}_b[v]$: backward node embedding
- $\mathbf{Y}[r]$: attribute embedding

PANE Algorithm

Forward & Backward Affinity Approximation

- Let $\mathbf{R}_\mathrm{r}$ be the row-normalization of attribute matrix $\mathbf{R}$. Then, compute

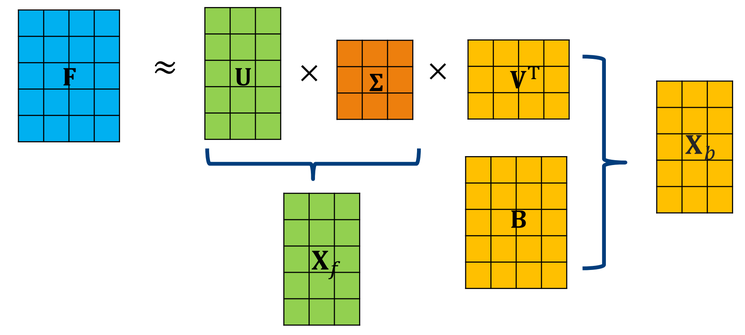
$$\mathbf{F} = \log\left(n \cdot \mathbf{\Pi}_f + 1\right), \mathbf{\Pi}_f = \mathrm{cnormalize}\left(\alpha \sum_{i=0}^{t} (1-\alpha)^i \mathbf{P}^i \cdot \mathbf{R}_\mathrm{r}\right)$$

- Let $\mathbf{R}_\mathrm{c}$ be the column-normalization of attribute matrix $\mathbf{R}$. Then, compute

$$\mathbf{B} = \log\left(d \cdot \mathbf{\Pi}_b + 1\right), \mathbf{\Pi}_b = \mathrm{rnormalize}\left(\alpha \sum_{i=0}^{t} (1-\alpha)^i \mathbf{P}^i \cdot \mathbf{R}_\mathrm{c}\right)$$

Greedy Initialization

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



Coordinate Descent

- For t iterations
- For $v_i \in V, l \in [1, k/2]$

$$\mu_f(v_i, l) = \frac{(\mathbf{X}_f \mathbf{Y}^\mathrm{T} - \mathbf{F})[v_i] \cdot \mathbf{Y}[:, l]}{\mathbf{Y}^\mathrm{T}[l] \cdot \mathbf{Y}[:, l]}, \mathbf{X}_f[v_i, l] = \mathbf{X}_f[v_i, l] - \mu_f(v_i, l)$$

$$\mu_b(v_i, l) = \frac{(\mathbf{X}_b \mathbf{Y}^\mathrm{T} - \mathbf{F})[v_i] \cdot \mathbf{Y}[:, l]}{\mathbf{Y}^\mathrm{T}[l] \cdot \mathbf{Y}[:, l]}, \mathbf{X}_b[v_i, l] = \mathbf{X}_b[v_i, l] - \mu_b(v_i, l)$$

- For $r_j \in R, l \in [1, k/2]$

$$\mu_y(r_j, l) = \frac{\mathbf{X}_f^T[l] \cdot (\mathbf{X}_f \mathbf{Y}^\mathrm{T} - \mathbf{F})[:, r_j] + \mathbf{X}_b^T[l] \cdot (\mathbf{X}_b \mathbf{Y}^\mathrm{T} - \mathbf{B})[:, r_j]}{\mathbf{X}_f^T[l] \cdot \mathbf{X}_f[:, l] + \mathbf{X}_b^T[l] \cdot \mathbf{X}_b[:, l]}, \mathbf{Y}[r_j, l] = \mathbf{Y}[r_j, l] - \mu_y(r_j, l)$$

Experiments

Datasets:

Table 3: Datasets. (K=10^3, M=10^6)

Name	$\lvert V \rvert$	$\lvert E_V \rvert$	$\lvert R \rvert$	$\lvert E_R \rvert$	$\lvert L \rvert$	Refs
Cora	2.7K	5.4K	1.4K	49.2K	7	[25, 27, 30, 41, 44, 51]
Citeseer	3.3K	4.7K	3.7K	105.2K	6	[25, 27, 30, 41, 44, 51]
Facebook	4K	88.2K	1.3K	33.3K	193	[24, 27, 45, 49]
Pubmed	19.7K	44.3K	0.5K	988K	3	[27, 30, 49, 51]
Flickr	7.6K	479.5K	12.1K	182.5K	9	[27]
Google+	107.6K	13.7M	15.9K	300.6M	468	[24, 45]
TWeibo	2.3M	50.7M	1.7K	16.8M	8	-
MAG	59.3M	978.2M	2K	434.4M	100	-

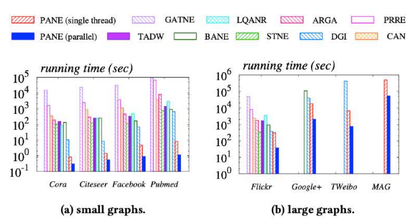
NN-based methods
1. STNE [KDD 2018]
2. ARGA [IJCAI 2018]
3. LQANR [IJCAI 2019]
4. CAN [WSDM 2019]
5. DGI [ICLR 2019]
6. GATNE [KDD 2019]

MF-based methods
1. TADW [IJCAI 2015]
2. BANE [ICDM 2018]
3. NRP [VLDB 2020]

Other method
1. PRRE [CIKM 2018]

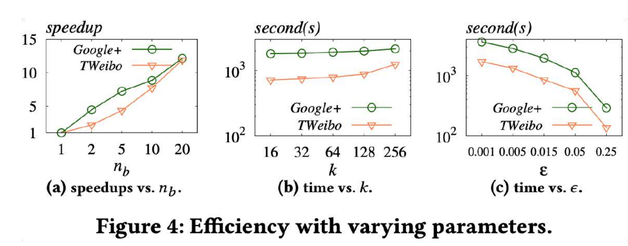
Experiments for efficiency and scalability

- Outperforms competitors often by orders of magnitude



(a) small graphs. (b) large graphs.

Figure 3: Running time (best viewed in color).



(a) speedups vs. n_b. (b) time vs. k. (c) time vs. ϵ.

Figure 4: Efficiency with varying parameters.

Experiments for attribute inference, link prediction & node classification

- Consistently achieve the best attribute inference performance on all datasets and significantly outperforms existing solutions by a large margin.
- Outperform all competitors over all datasets except NRP on Google+, by a substantial margin of up to 6.6% for AUC and up to 13% for AP.
- Outperform competitors by 3.4%-17.2% on node classification.

Table 4: Attribute inference performance.

Method	*Cora*		*Citeseer*		*Facebook*		*Pubmed*		*Flickr*		*Google+*		*TWeibo*		*MAG*	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
BLA	0.559	0.563	0.540	0.541	0.653	0.648	0.520	0.524	0.660	0.653	-	-	-	-	-	-
CAN	0.865	0.855	0.875	0.859	0.765	0.745	0.734	0.72	0.772	0.774	-	-	-	-	-	-
PANE (single thread)	0.913	0.925	0.903	0.916	0.828	0.84	0.871	0.874	0.825	0.832	0.972	0.973	0.774	0.837	0.876	0.888
PANE (parallel)	0.909	0.92	0.899	0.913	0.825	0.837	0.867	0.869	0.822	0.831	0.969	0.97	0.773	0.836	0.874	0.887

Table 5: Link prediction performance.

Method	*Cora*		*Citeseer*		*Pubmed*		*Facebook*		*Flickr*		*Google+*		*TWeibo*		*MAG*	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
NRP	0.796	0.777	0.86	0.808	0.87	0.861	0.969	0.973	0.909	0.902	0.989	0.992	0.967	0.979	0.915	0.92
GATNE	0.791	0.822	0.687	0.767	0.745	0.796	0.961	0.954	0.805	0.785	-	-	-	-	-	-
TADW	0.829	0.805	0.895	0.868	0.904	0.863	0.752	0.793	0.573	0.58	-	-	-	-	-	-
ARGA	0.64	0.485	0.637	0.484	0.623	0.474	0.71	0.636	0.676	0.656	-	-	-	-	-	-
BANE	0.875	0.823	0.899	0.873	0.919	0.847	0.796	0.795	0.64	0.605	0.56	0.533	-	-	-	-
PRRE	0.879	0.836	0.895	0.855	0.887	0.813	0.899	0.884	0.789	0.806	-	-	-	-	-	-
STNE	0.808	0.829	0.71	0.781	0.789	0.774	0.962	0.957	0.638	0.659	-	-	-	-	-	-
CAN	0.663	0.559	0.734	0.652	0.734	0.559	0.714	0.639	0.5	0.5	-	-	-	-	-	-
DGI	0.51	0.4	0.5	0.4	0.73	0.554	0.711	0.637	0.769	0.824	0.792	0.795	0.721	0.64	-	-
LQANR	0.886	0.863	0.916	0.916	0.904	0.8	0.951	0.917	0.824	0.805	-	-	-	-	-	-
PANE (single thread)	0.933	0.918	0.932	0.919	0.985	0.977	0.982	0.982	0.929	0.927	0.987	0.982	0.976	0.986	0.96	0.965
PANE (parallel)	0.929	0.914	0.929	0.916	0.985	0.976	0.98	0.979	0.927	0.924	0.984	0.98	0.975	0.985	0.958	0.962

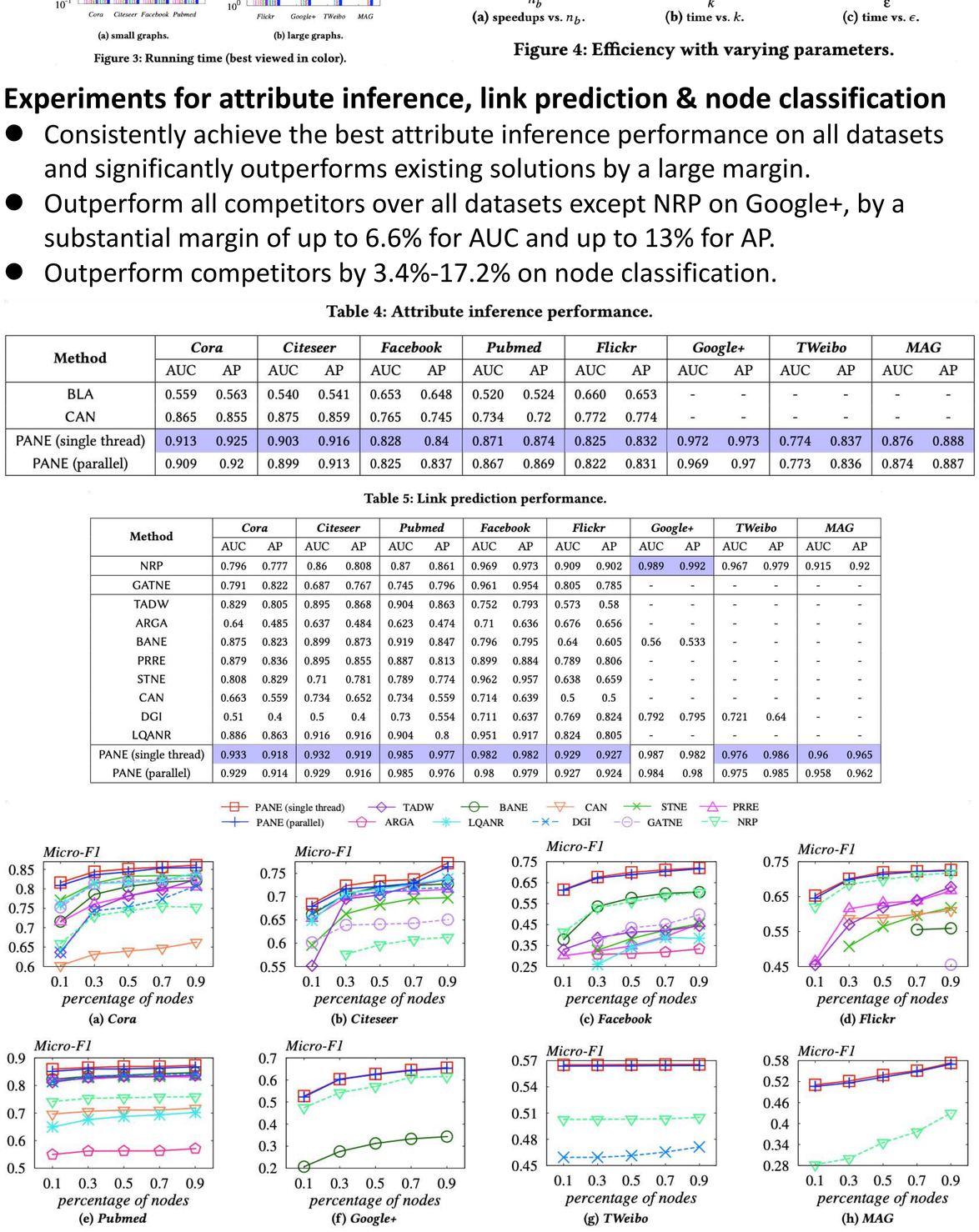


Figure 2: Node classification results (best viewed in color).

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