Machine Learning with Graphs (MLG)

Learning to Identify High Betweenness Nodes

Cheng-Te Li (李政德)

Institute of Data Science

National Cheng Kung University

chengte@mail.ncku.edu.tw

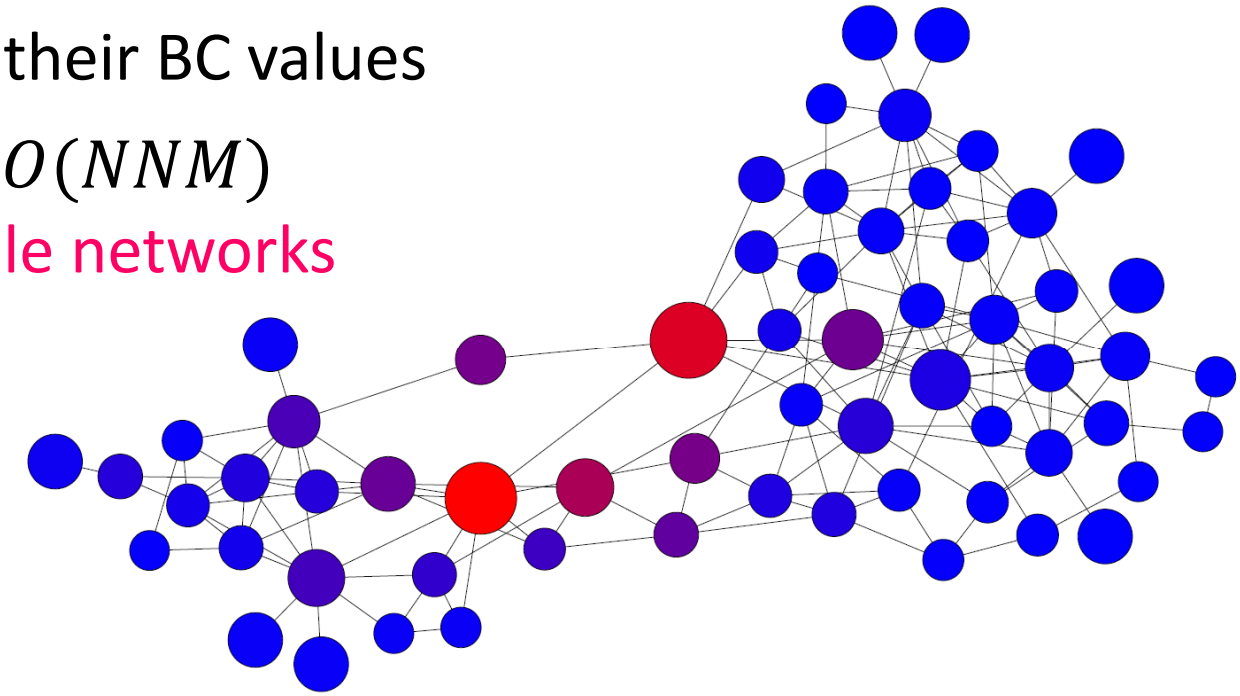
Identify High BC Nodes

- Applications of Finding High BC nodes
  - Spreading novel information
  - Bottleneck of power grid
  - Blocking the propagation of misinformation
  - Cross-disciplinary papers

(1) Calculate BC for all nodes
(2) Rank all nodes based on their BC values

Calculating BC for all nodes: $O(NNM)$
→ unaffordable for large-scale networks

N: number of nodes
M: number of edges



Deep Ranker for BC

- Transform the problem of identifying high BC nodes into a learning problem
- Idea: map each node in a graph into a ranking score that reflects its BC
  - Need not to approximate exact BC values
  - But indicate the relative order of nodes w.r.t BC
- An **encoder-decoder** framework
  - Encoder: map each node to an embedding vector that captures structural info
  - Decoder: map embedding vectors into BC ranking scores

DrBC: NN-based Model

Encoder

- Initial feature $X_v = [d_v, 1, 1]$
- Neighborhood $N(v)$:
  all immediate neighbors
  of a node v in the graph
- Neighborhood Aggregation

$$h_{N(v)}^{(l)} = \sum_{j \in N(v)} \frac{1}{\sqrt{d_v + 1}\sqrt{d_j + 1}} h_j^{(l-1)}$$

- COMBINE Function

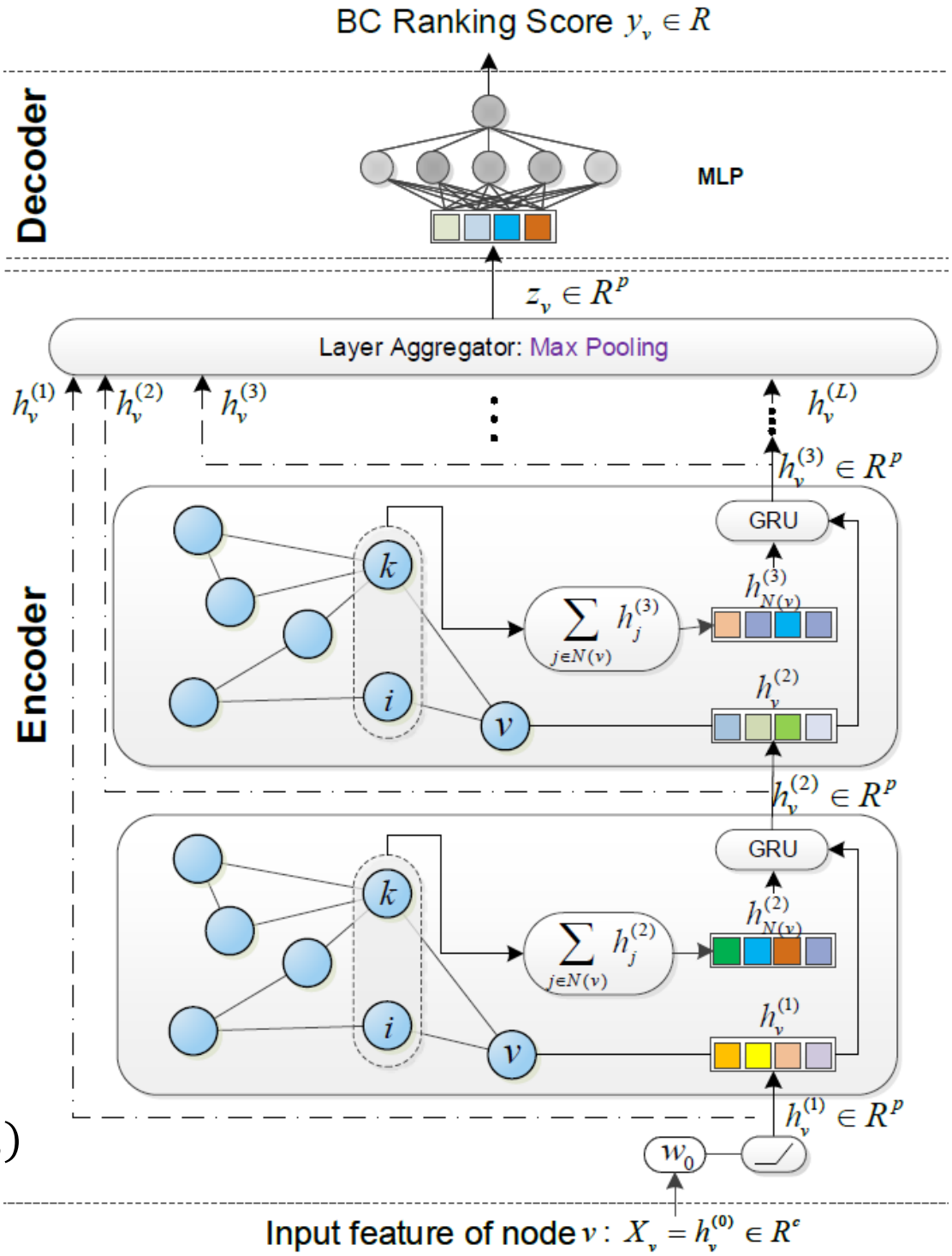
$$h_v^{(l)} = GRUCell\left(h_v^{(l-1)}, h_{N(v)}^{(l)}\right)$$

- Layer Aggregation

$$z_v = \max\left(h_v^{(1)}, h_v^{(2)}, \ldots, h_v^{(L)}\right)$$

MLP Decoder $y_v = W_5 \cdot ReLU(W_4 z_v)$

- Two-layer neural networks



DrBC Encoder

Algorithm 1 DrBC encoder function

Input: Network $G = (V, E)$; input features $\{X_v \in \mathbb{R}^c, \forall v \in V\}$; depth L; weight matrices $W_0, W_1, U_1, W_2, U_2, W_3, U_3$.

Output: Vector representations $z_v, \forall v \in V$.

Initialize $h_v^{(0)} = X_v$;

$h_v^{(1)} = ReLU(W_0 h_v^{(0)}), h_v^{(1)} = h_v^{(1)} / \|h_v^{(1)}\|_2, \forall v \in V$;

for $l = 2$ to L **do**

  for $v \in V$ **do**

    $h_{N(v)}^{(l)} = \sum_{j \in N(v)} \frac{1}{\sqrt{d_v+1} \cdot \sqrt{d_j+1}} h_j^{(l-1)}$;

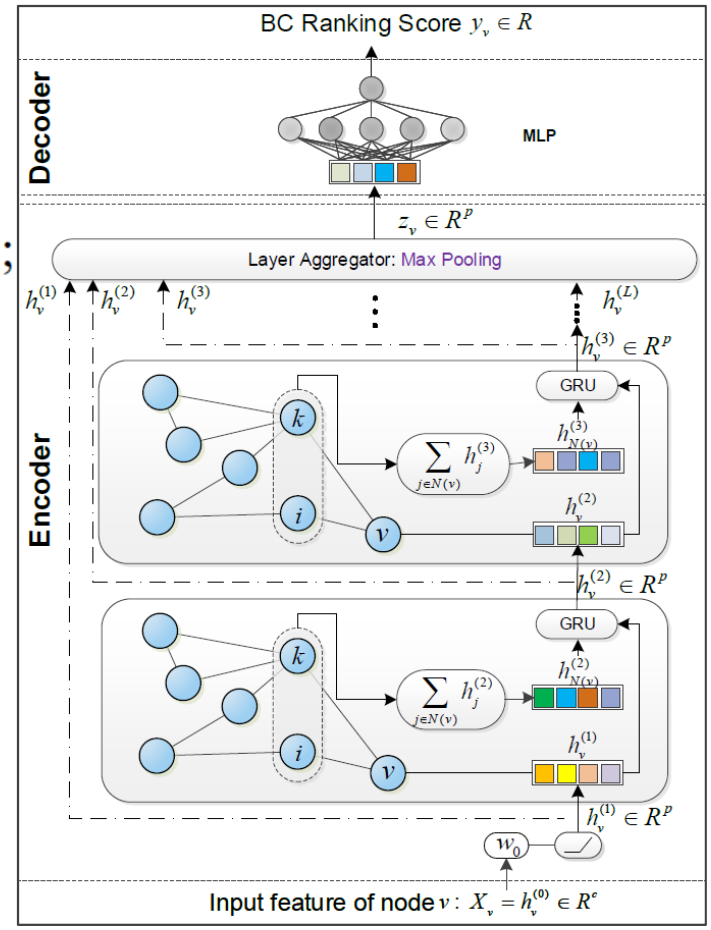
    $h_v^{(l)} = GRUCell(h_v^{(l-1)}, h_{N(v)}^{(l)})$;

  end for

  $h_v^{(l)} = h_v^{(l)} / \|h_v^{(l)}\|_2, \forall v \in V$;

end for

$z_v = max(h_v^{(1)}, h_v^{(2)}, \ldots, h_v^{(L)}), \forall v \in V$;



Training Algorithm

- Pairwise Ranking Loss
  - Sample a node pair (i, j)
  - Compute its difference of true BC values, $b_{ij} = b_i - b_j$
  - Ranking score difference: $y_{ij} = y_i - y_j$

$$L = \sum_{(i,j)\in P} -\sigma(b_{ij}) \log\left(\sigma(y_{ij})\right) - \left(1 - \sigma(b_{ij})\right) \log\left(1 - \sigma(y_{ij})\right)$$

P: a set of sampled node pairs

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Inductive Learning
  - Train the model on randomly generated synthetic graphs
  - Apply the trained model to other synthetic and real graphs

DrBC Training

Algorithm 2 Training algorithm for DrBC

Input: Encoder parameters $\Theta_{ENC} = (W_0, W_1, U_1, W_2, U_2, W_3, U_3)$, Decoder parameters $\Theta_{DEC} = (W_4, W_5)$

Output: Trained Model M

1: **for** each episode **do** // Generating synthetic graph
2: Draw network G from distribution D (like the power-law model)
3: Calculate each node's exact BC value $b_v, \forall v \in V$ // training target
4: Get each node's embedding $z_v, \forall v \in V$ with Algorithm 1 // encoder
5: Compute BC ranking score y_v for each node v with $y_v = W_5 \cdot ReLU(W_4 z_v)$
6: Sample source nodes and target nodes, and form a batch of node pairs
7: Update $\Theta = (\Theta_{ENC}, \Theta_{DEC})$ with Adam by minimizing Loss L
8: **end for**

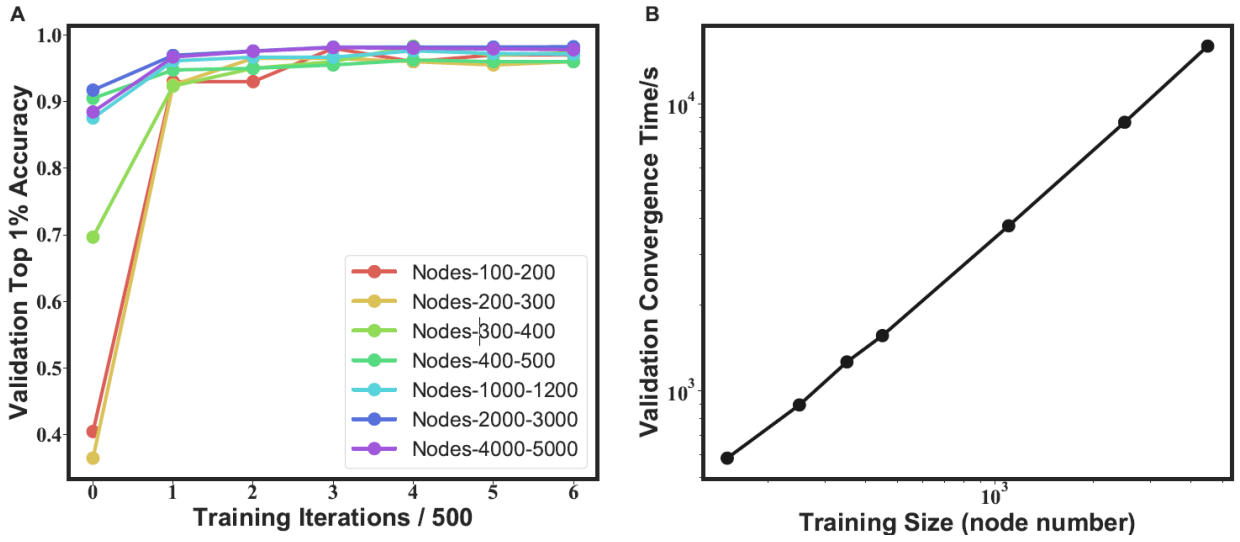
// Sampling node pairs

Graph Generation & Pair Sampling

- Generating a synthetic graph for training
  - Generating graphs by the power-law cluster model with n="number of nodes", m=4, p=0.05
  - **networkx**.generators.random_graphs.**powerlaw_cluster_graph**

    https://networkx.github.io/documentation/stable/reference/generated/networkx.generators.random_graphs.powerlaw_cluster_graph.html#networkx.generators.random_graphs.powerlaw_cluster_graph

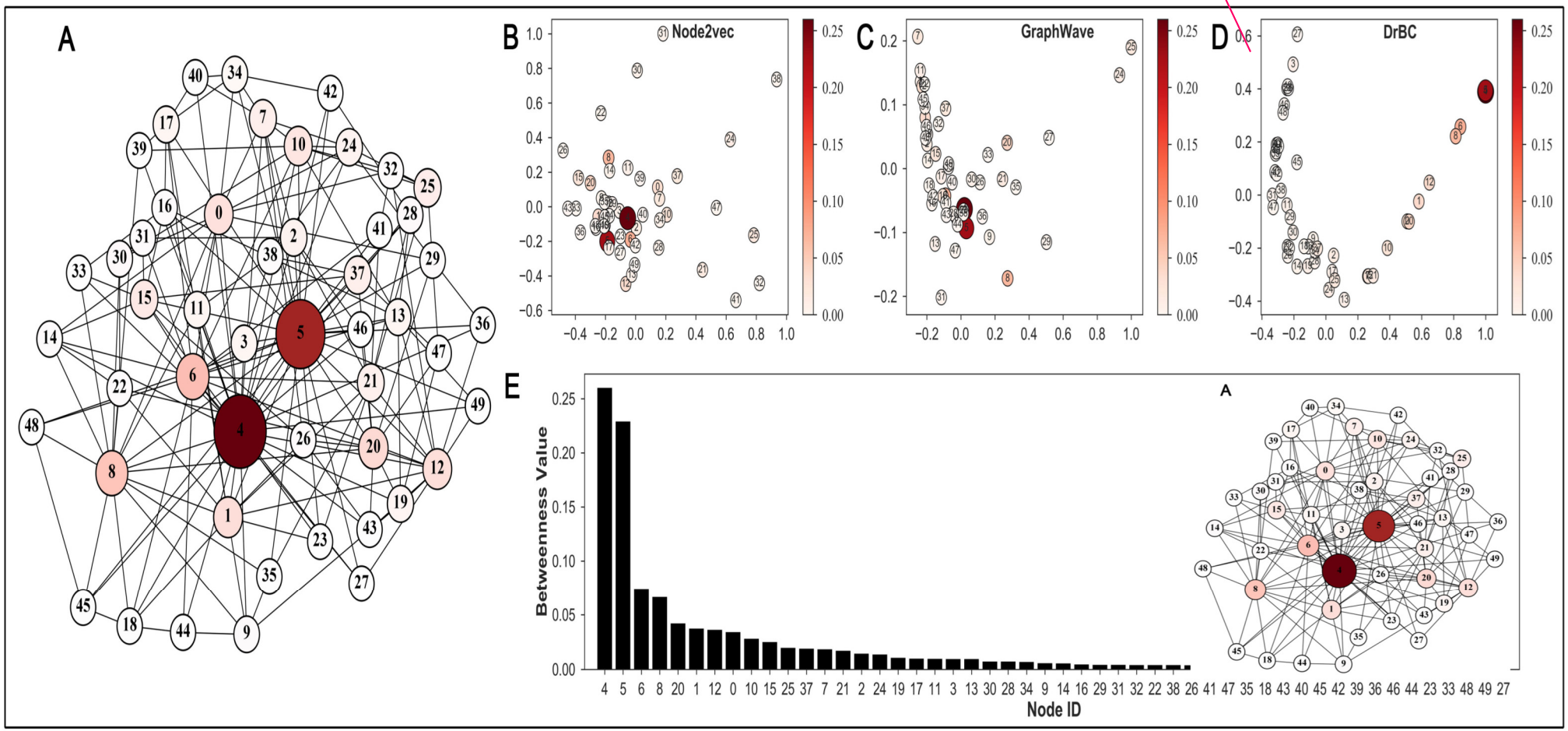
- Sampling node pairs from the generated graph
  - Sampling source and target nodes with replacement
  - 5|V| node pairs in total
  - Convergence time increases linearly with training scale
  - Training scale of 4000-5000 is satisfying and adopted



Results on Synthetic Graph

2D PCA projection to visualize the learned embeddings to show the model preserves the relative BC order between nodes in the embedding space

Only DrBC leads to the linear separable portions correspond to clusters of nodes with similar distribution of true BC



Performance & Efficiency on Large-scale Real Graphs

Network	|V|	|E|
com-Youtube	1,134,890	2,987,624
Amazon	2,146,057	5,743,146
Dblp	4,000,148	8,649,011
cit-Patents	3,764,117	16,511,741
com-lj	3,997,962	34,681,189

Network	Top-1%					Top-5%				
	ABRA	RK	KADABRA	Node2Vec	DrBC	ABRA	RK	KADABRA	Node2Vec	DrBC
com-youtube	**95.7**	76.0	57.5	12.3	73.6	**91.2**	75.8	47.3	18.9	66.7
amazon	69.2	86.0	47.6	16.7	**86.2**	58.0	59.4	56.0	23.2	**79.7**
Dblp	49.7	NA	35.2	11.5	**78.9**	45.5	NA	42.6	20.2	**72.0**
cit-Patents	37.0	**74.4**	23.4	0.04	48.3	42.4	**68.2**	25.1	0.29	57.5
com-lj	60.0	54.2*	31.9	3.9	**67.2**	56.9	NA	39.5	10.35	**72.6**

Network	Top-10%					Time/s				
	ABRA	RK	KADABRA	Node2Vec	DrBC	ABRA	RK	KADABRA	Node2Vec	DrBC
com-youtube	89.5	**100.0**	44.6	23.6	69.5	72898.7	125651.2	**116.1**	4729.8	402.9
amazon	60.3	**100.0**	56.7	26.6	76.9	5402.3	149680.6	**244.7**	10679.0	449.8
Dblp	**100.0**	NA	50.4	27.7	72.5	11591.5	NA	**398.1**	17446.9	566.7
cit-Patents	50.9	53.5	21.6	0.99	**64.1**	10704.6	252028.5	**568.0**	11729.1	744.1
com-lj	63.6	NA	47.6	15.4	**74.8**	34309.6	NA	**612.9**	18253.6	2274.2

DrBC Implications

- Neural network-based learning on graphs can be used to efficiently approximate high-complexity network analysis measures
  - E.g., DrBC for betweenness
- Can NN-based learning be used to find nodes with higher scores of other network-analysis measures?
  - High closeness nodes
  - High clustering coefficient (CC) node
  - High PageRank nodes
- The inductive learning capability of DrBC
  - Transfer the model learning from one graph to another!