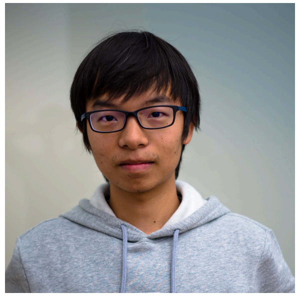
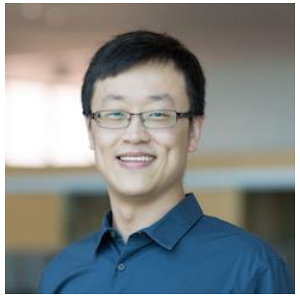
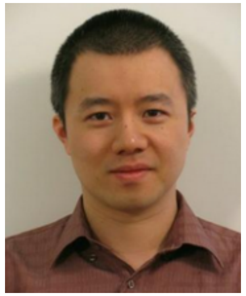
A Framework for Differentiable Discovery Of Graph Algorithms

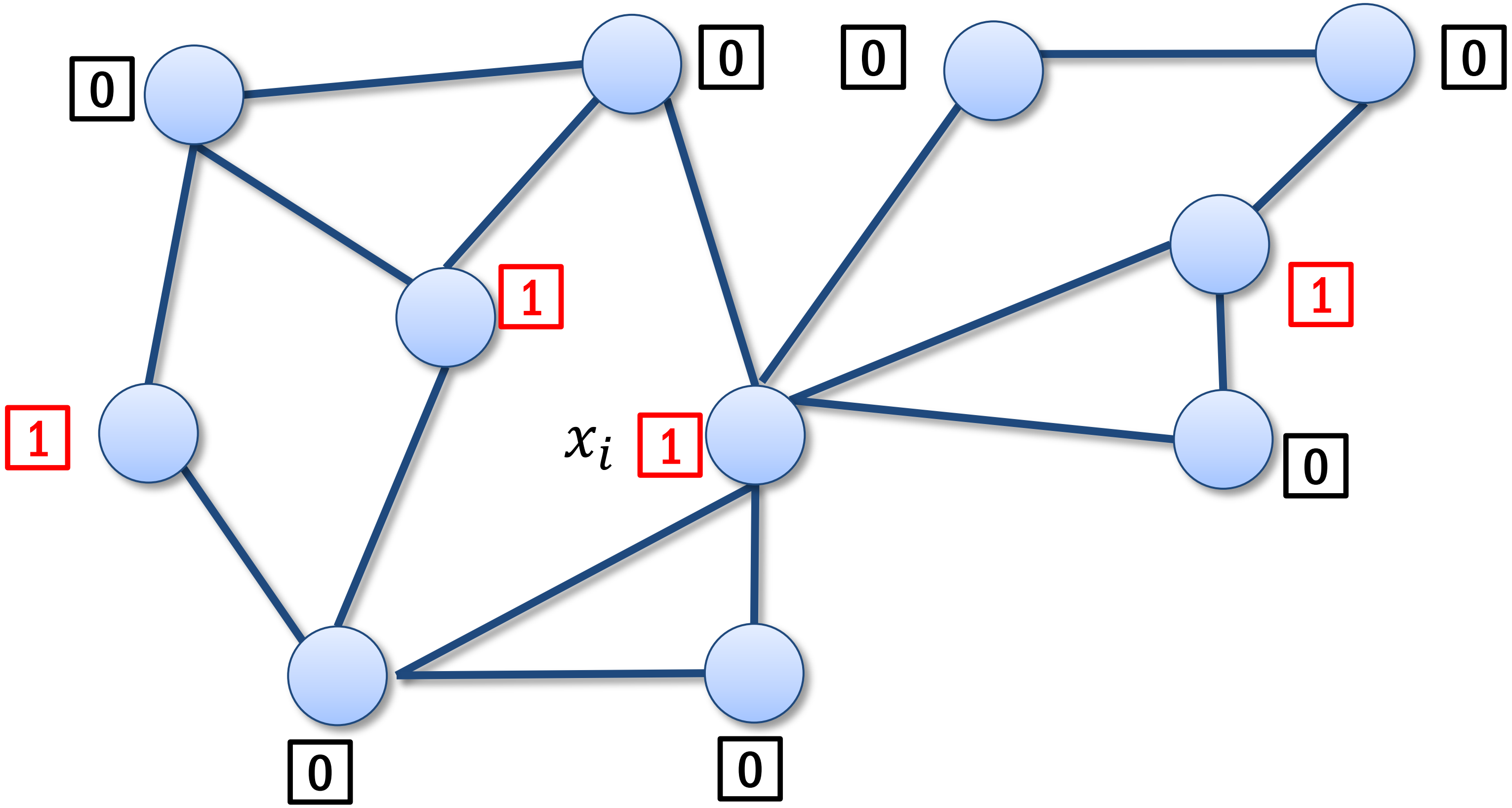
Hanjun Dai, Xinshi Chen, Yu Li, Xin Gao and Le Song



Google Research & Georgia Tech & KAUST

Combinatorial optimization over graph



Minimum vertex cover

$$\min_{x_i \in \{0,1\}} \sum_{i \in \mathcal{V}} x_i$$
$$\text{s.t. } x_i + x_j \geq 1, \forall (i,j) \in \mathcal{E}$$

NP-hard problems

2 - approximation algorithm for minimum vertex cover

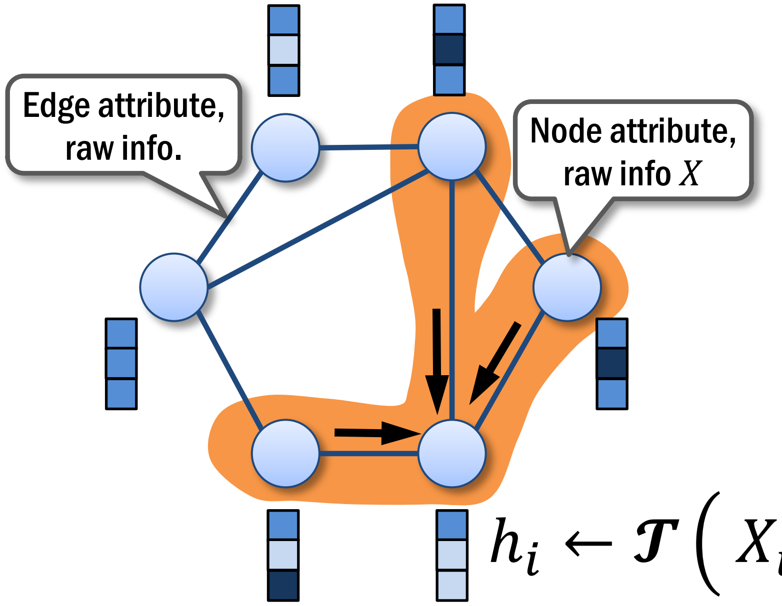
Repeat till all edges covered:

- Select uncovered edge with largest total degree

Manually designed policy. Can we learn from data?

GNN = Parametrized distributed local graph algorithm

Distributed Local Graph Algorithm =
Graph Representation + Iterative Update



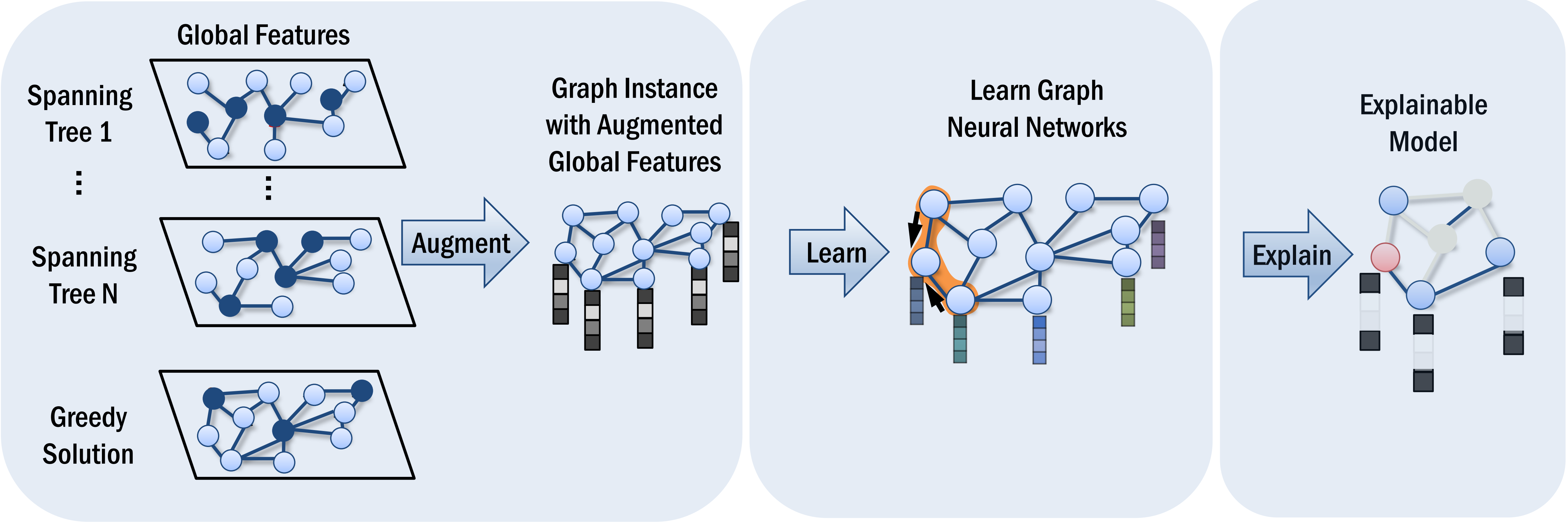
$$h_i \leftarrow \mathcal{T}\left(X_i, \{h_j\}_{j \in \mathcal{N}(i)} \right)$$

Differentiable Algorithm Discovery (DAD) framework

1. Can we discover new algorithms?

2. How to learn these algorithms?
Supervised, Unsupervised
Reinforcement Learning

3. Can we interpret what's discovered?



Search (Input) Space Design for GNN

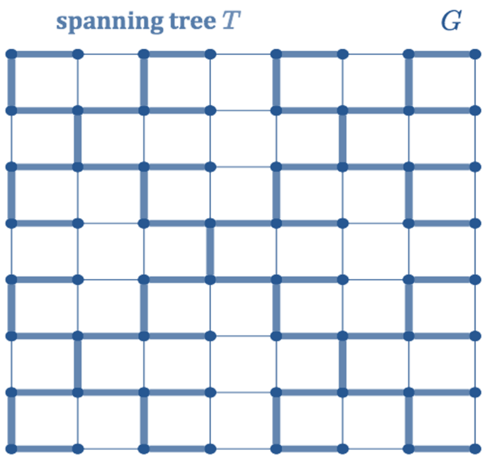
Motivating example

- The best known algorithm for solving a general linear system takes time $\mathcal{O}(n^{2.373})$
- Kelner et al. (2013) proposed an algorithm for solving Laplacian system:

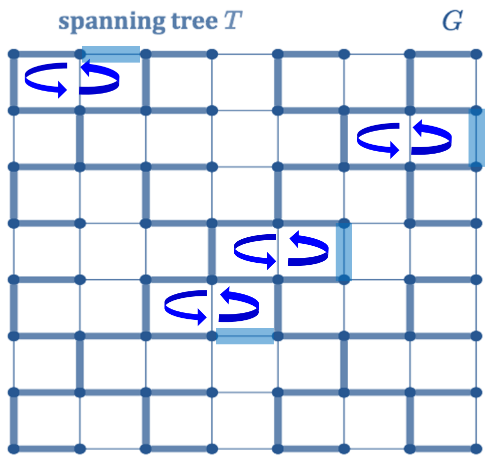
$$Lx = b, \text{ where } L \text{ is Laplacian matrix}$$

in nearly-linear time.

Step 1: Find a low-stretch spanning tree and obtain an initial solution on the tree.

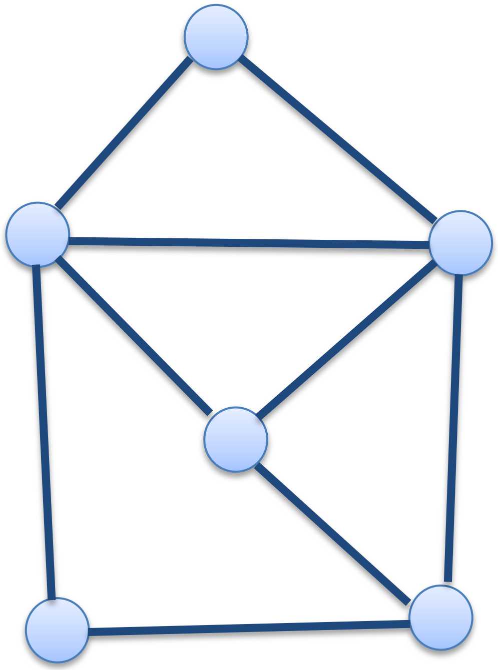


Step 2: Refine the initialized solution by iteratively operating on local cycles in the original graph.

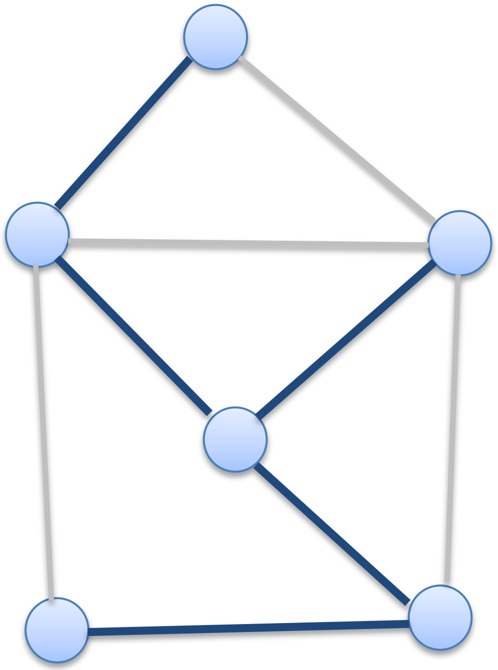


Cheap Solution as Global Features

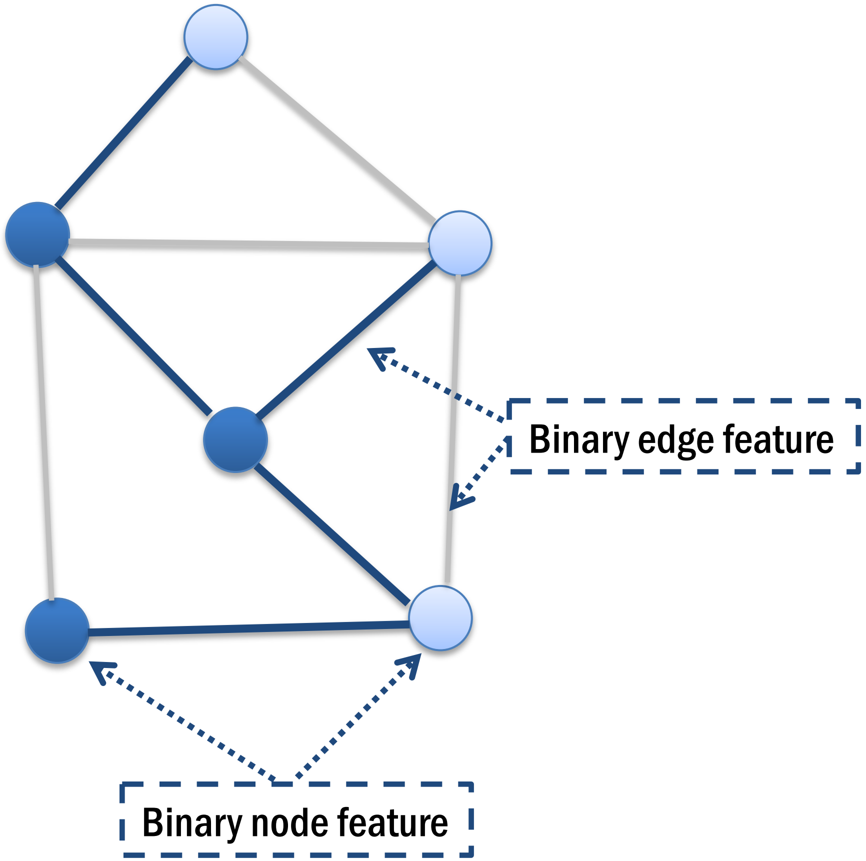
input graph G



spanning tree



optimal solutions y over tree
with dynamic programming

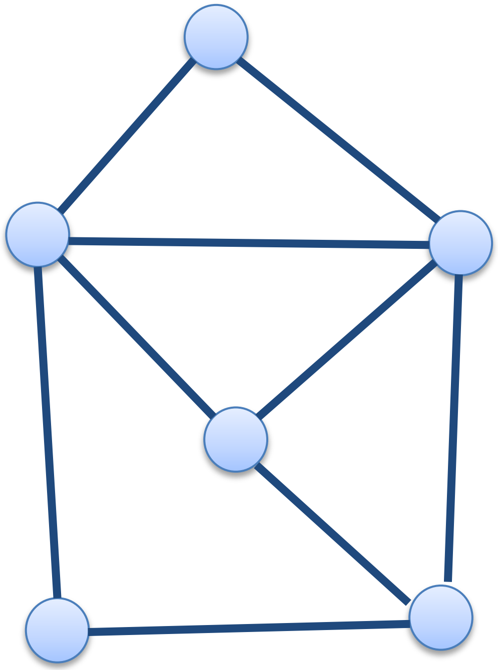


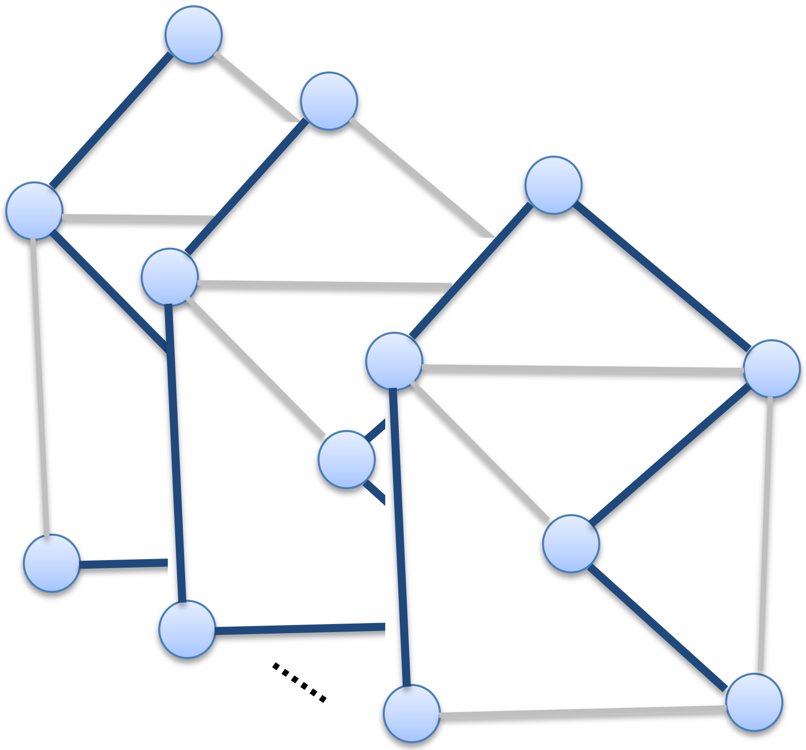
Cheap Solution as Global Features

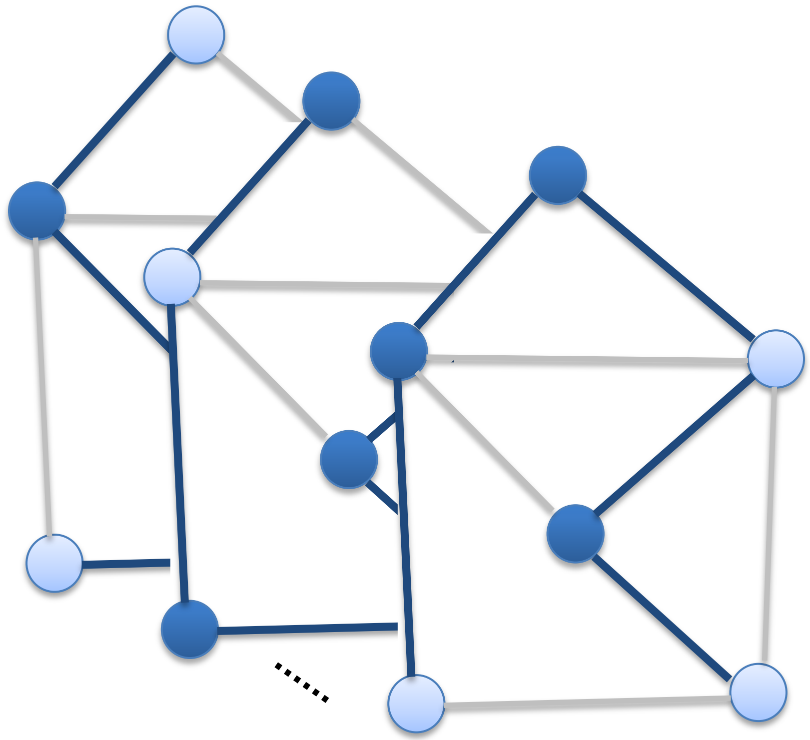
Input graph $\boxed{G}$

Spanning trees $\boxed{T(1), \ldots, T(n)}$

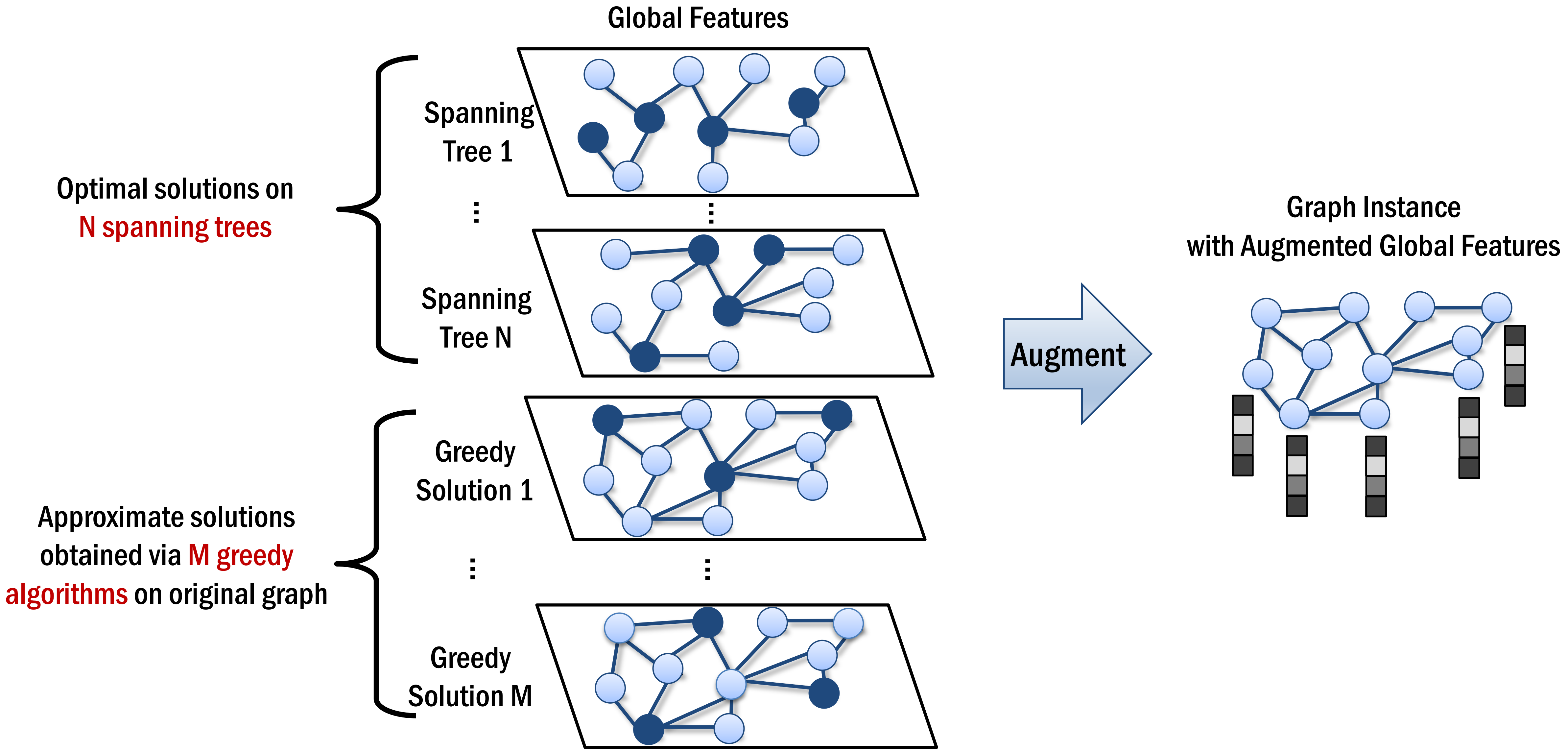
Optimal solutions over trees $\boxed{y^{T(1)}, \ldots, y^{T(n)}}$







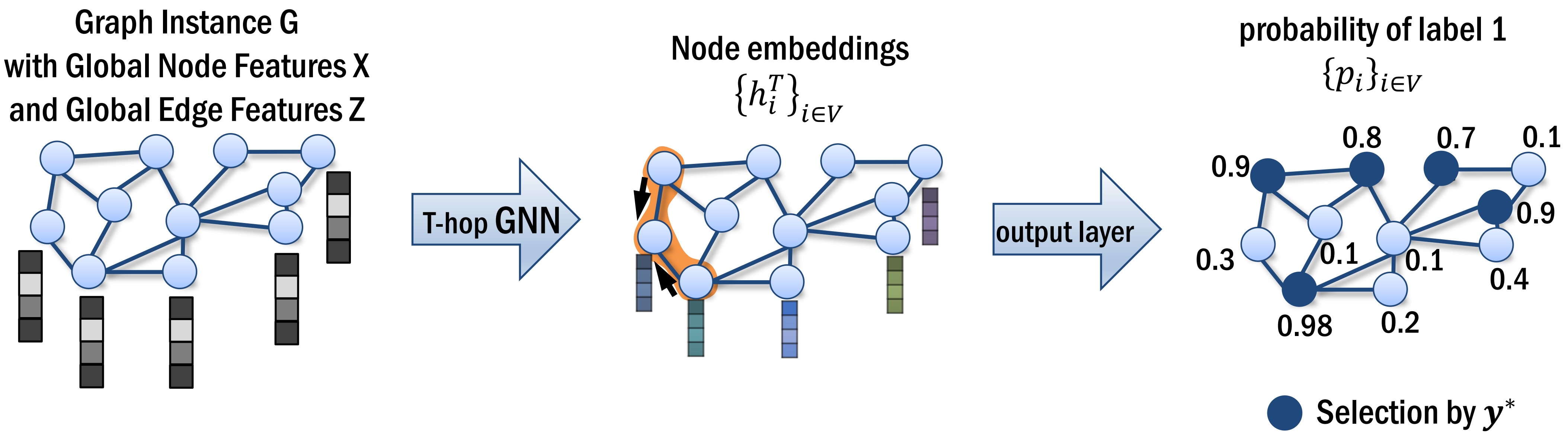
Cheap Solution as Global Features



Learning Local Iterative Algorithms with GNN

Supervised

- For each graph G, a solution y^* is obtained by running expensive solver
- Learn GNN-based algorithm which can imitate y^* but runs much faster



Unsupervised

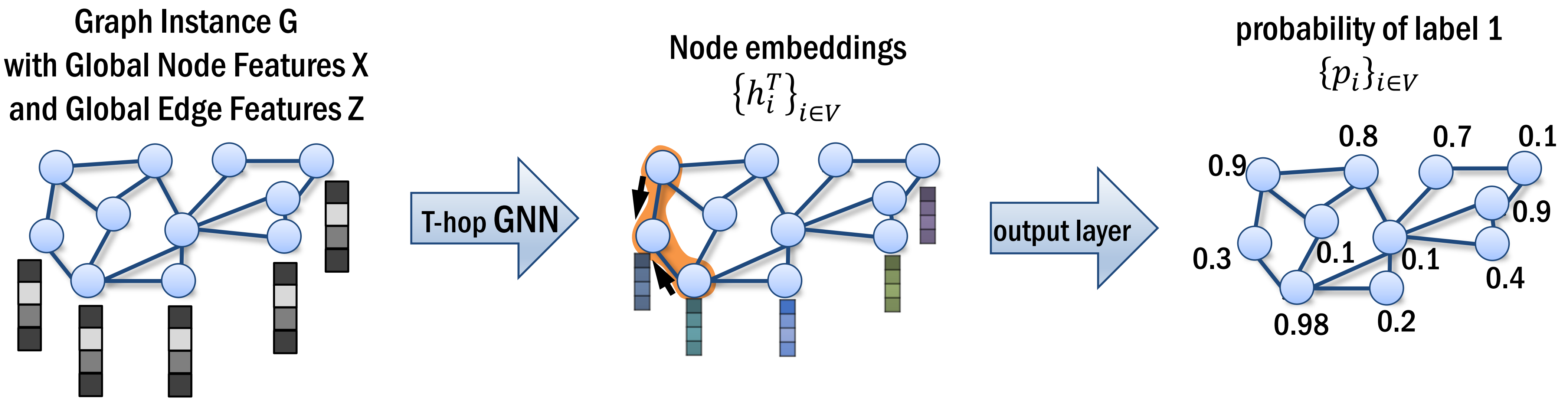
- Many graph problems can be formulated as integer programming (IP) problems:

$$\min_{y\in\{0,1\}^{|V|}} f(Y;G) \quad \text{subject to} \quad g_i(y;G) \leq 0 \ \text{ for } i = 1, \dots, l$$

- Construct unsupervised training loss based on optimization objective f and constraints g

$$L_U(p,G) \coloneqq E[f(Y;G)] + \beta \cdot P[g_i(Y;G) \leq 0]$$

where $Y \sim \text{Bernoulli}(p)$

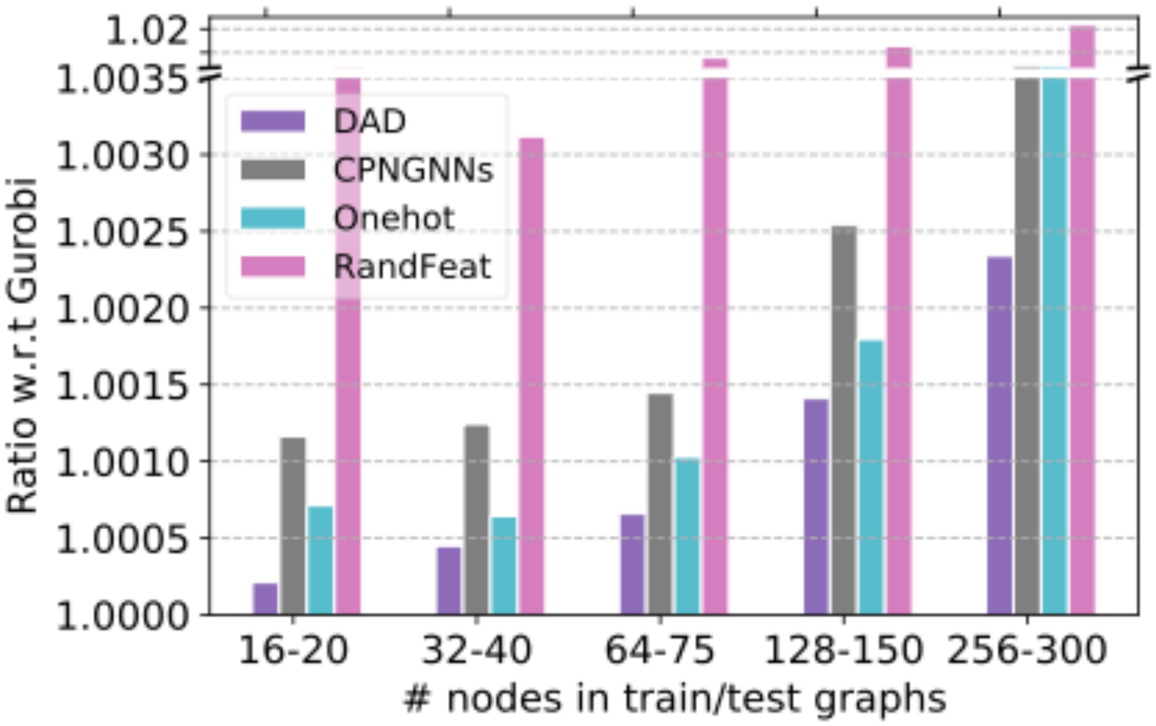


Better learned algorithms with global information

- Comparison of our features to other features
  - Random features
  - Random one-hot encoding
  - Port Numbering + Weak 2-coloring (CPNGNN)

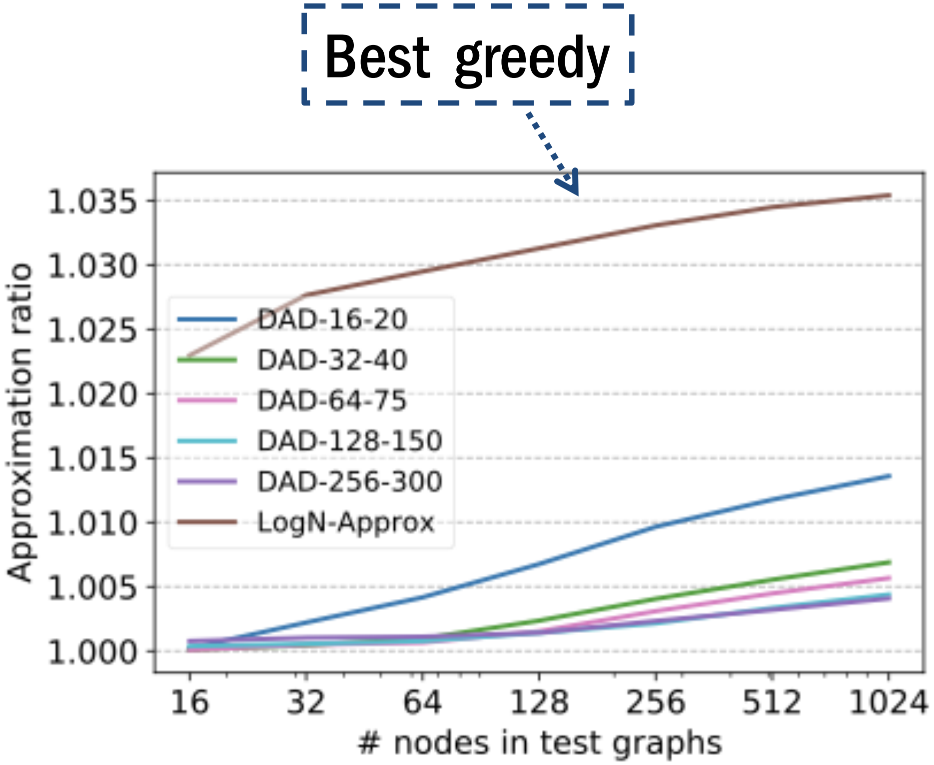
Our approach is consistently better

Minimum Vertex Cover



A deeper understanding of the performance

- Extrapolation
  - train on small graphs
  - test on graphs up to 1024 nodes

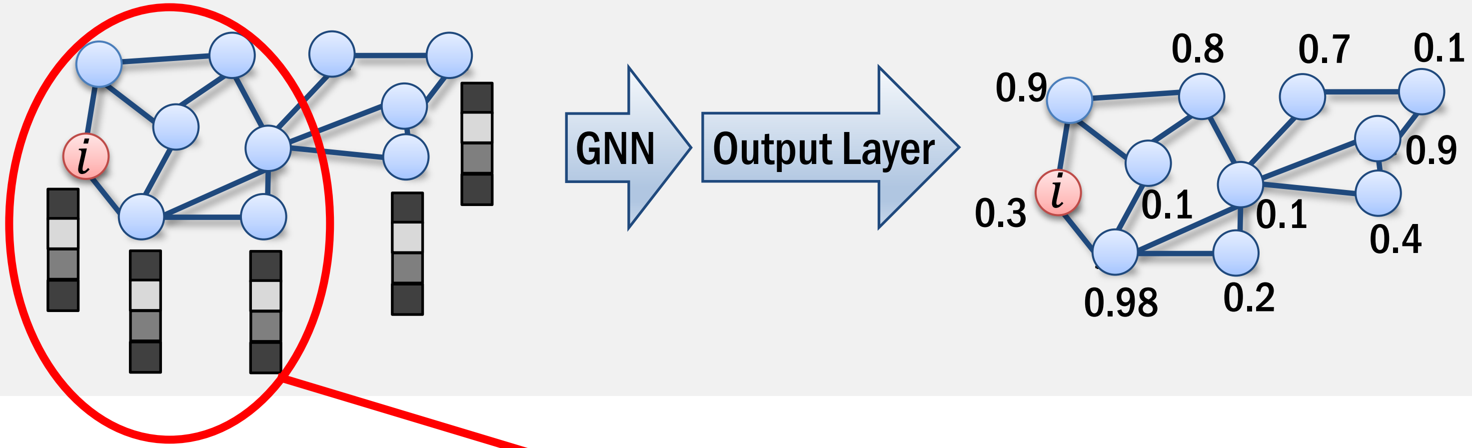


Minimum Vertex Cover on Barabasi Albert random graphs

Explain the Learned GNN Algorithms

Explainer

Learned Algorithm:



T-hop subgraph

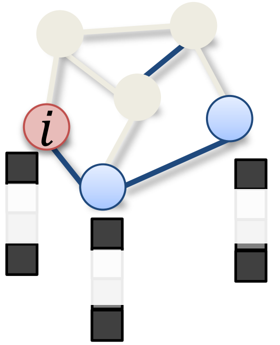
Node selection probability

$$p_i = \mathrm{Output}\left(h_i^{(T)}\right)$$

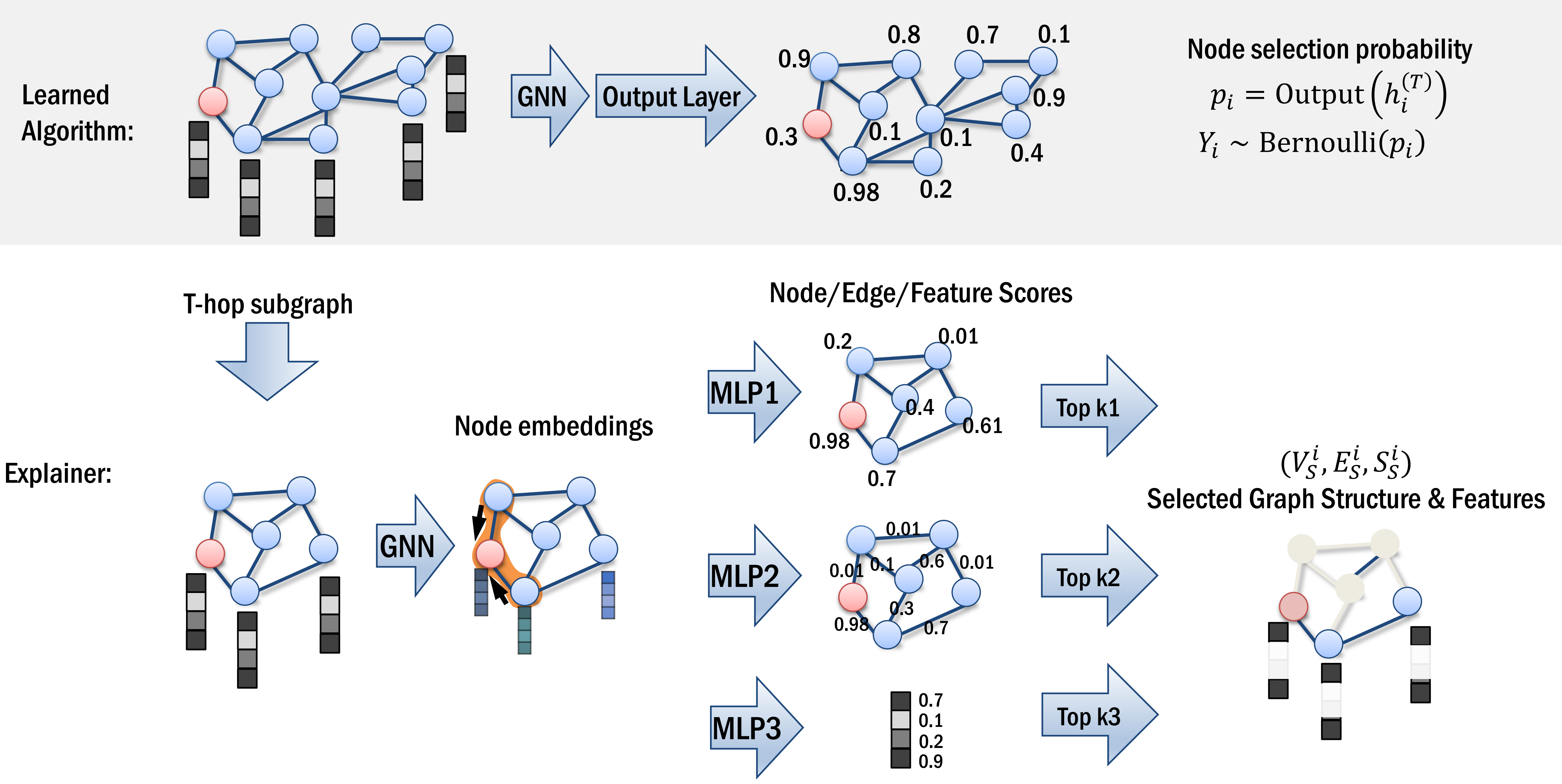
$$Y_i \sim \mathrm{Bernoulli}(p_i)$$

(V_S^i, E_S^i, S_S^i)

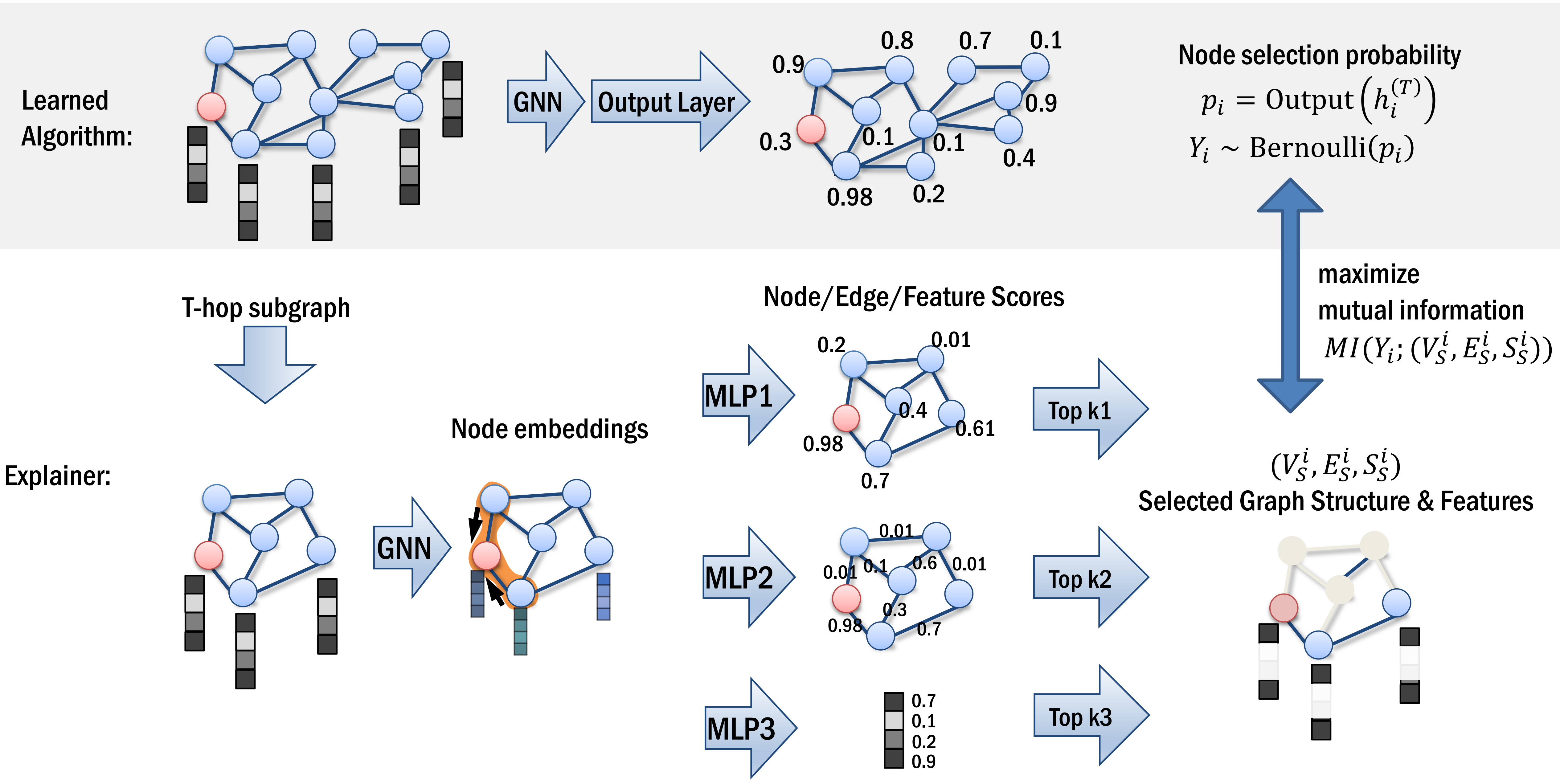
Selected Graph Structure & Features



Explainer architecture



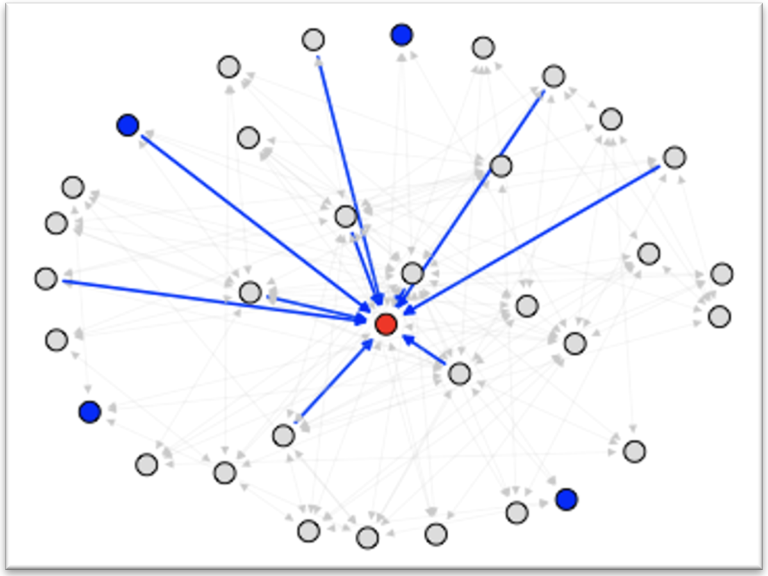
Information theoretic learning

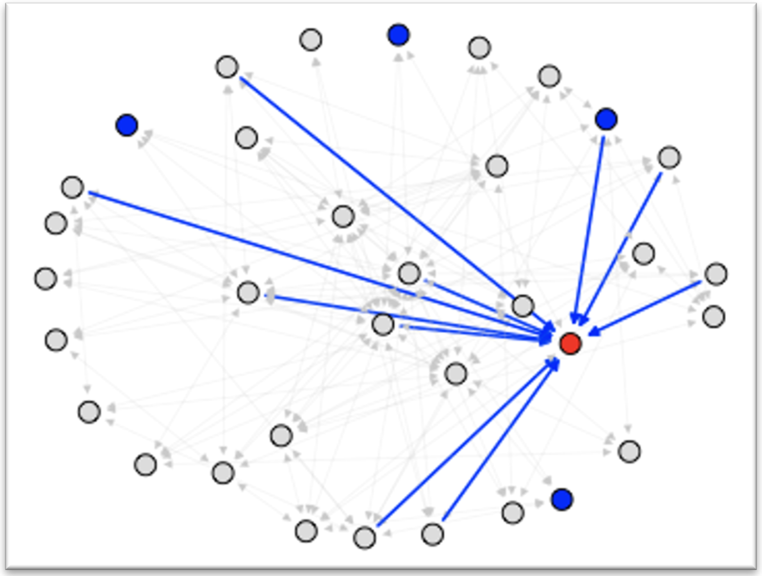


Discovery of greedy-like behavior

- Explanation setting:
  - limit to 5 nodes and 10 edges to explain each target node

Minimum
Vertex
Cover



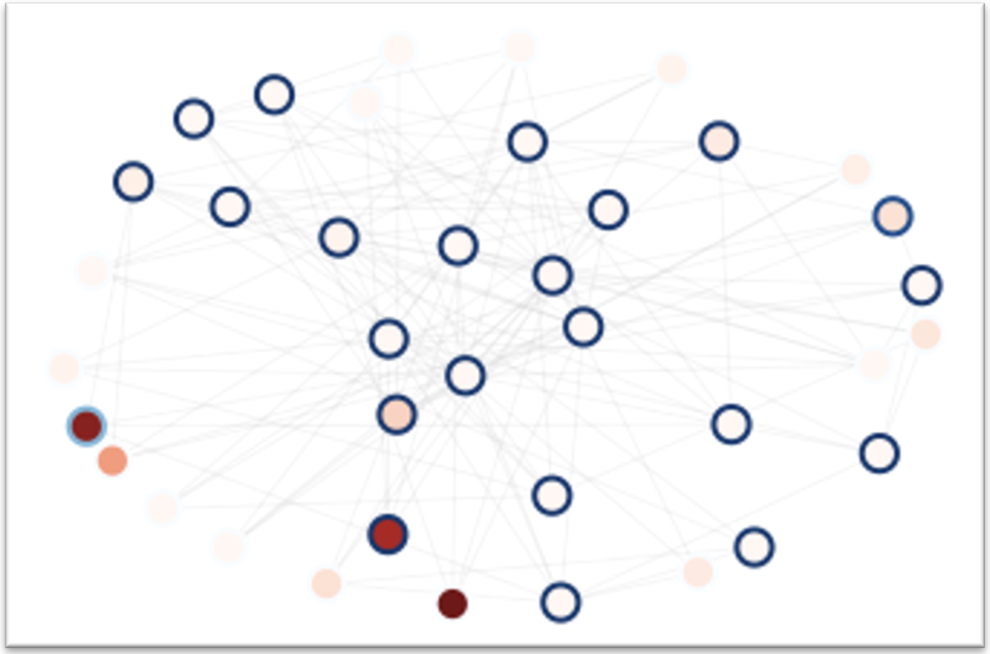


Greedy like behavior on some targets!

- Takeaway:
  - Greedy heuristics are the best performing ones on these tasks
  - GNN understands and learns the meaning of greedy algorithm features

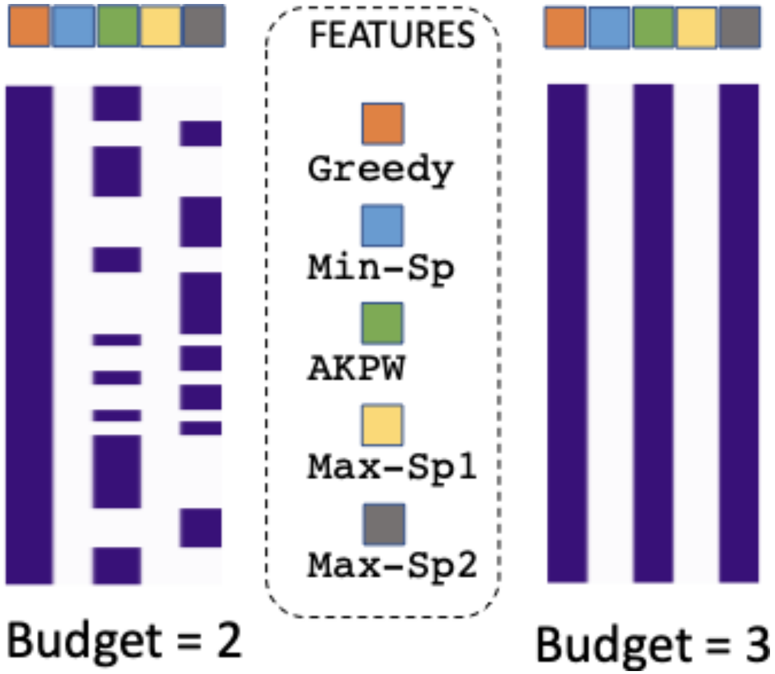
Discovery of anchor nodes

Minimum Vertex Cover



What global features are effective?

Max-Cut

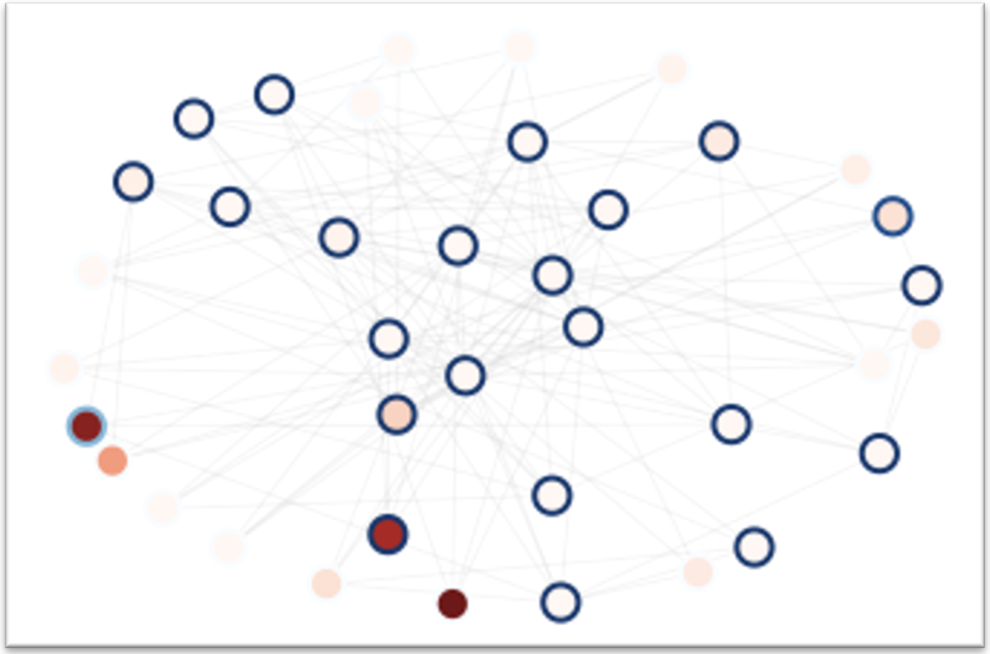


Q/A

Discovery of anchor nodes

- Node color: the darker, the more frequent of being selected for explanation

Minimum
Vertex
Cover



- Observation
  - There exists a set of "anchor nodes"
  - Anchor nodes tends to be diverse

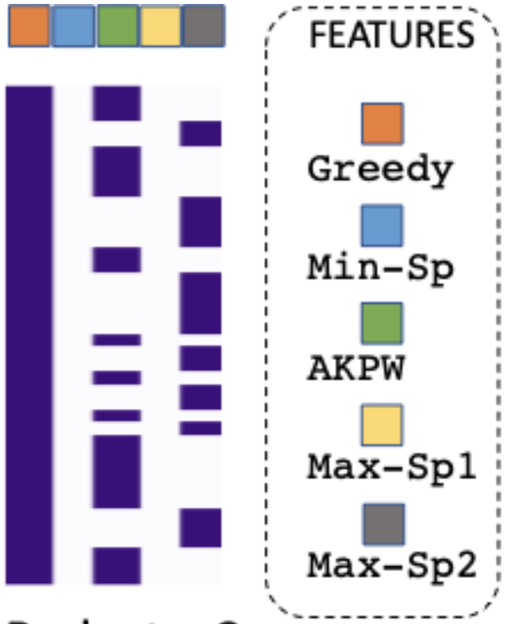
- Hypothesis
  - Anchor nodes are like "landmarks" in the graph
  - GNN compares the target node with anchor nodes to make prediction

- Connections: GNN with anchor nodes: Position/distance aware GNNs (You et.al, 2018; Li et.al, 2020)

What global features are effective?

- Explanation setting:
  - Max-cut problem
  - limit to 2 or 3 global features

- Budget=2
  - Greedy is always selected;
  - AKPW or max-spanning tree can be selected with equal chances;
  - Two max-spanning tree solutions will not be selected at the same time;



Budget = 2

- Budget=3
  - feature selection is consistent across different target nodes;
  - The {Greedy, AKPW, Max-Spanning} are the best performing three;
  - Again, two Max-spanning trees solutions will not appear at the same time, even though itself performs better than AKPW;