On Explainability of Graph Neural Networks via Subgraph Explorations

Chao Chen

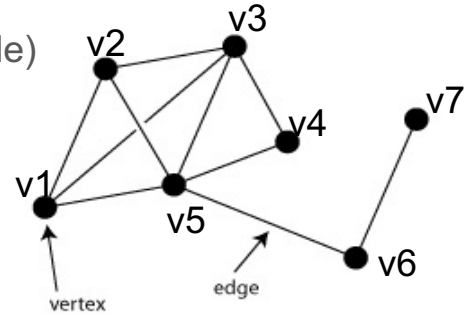
LEHIGH
UNIVERSITY

Scope of Explanations for GNN:

Node-level / Edge-level: GNNExplainer[1], PGExplainer[2] (explain a target node)

Highlight important edges or nodes.

Cons: not guarantee to be connected, ignore interactions within graphs.



Model-level: XGNN[3] (explain a target model)

Extract the most important patterns for **model's** prediction w.r.t. a specific class c_i

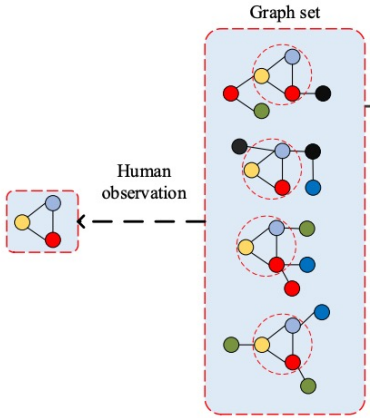
$$G^* = \arg\max_G P(f(G) = c_i)$$

Cons: not input-dependent, less precise

Subgraph-level: SubgraphX (explain a target node or graph)

Extract a subgraph for target graph's prediction

$$G^* = \arg \max_{|G_i| \le N_{min}} Score(f, G, G_i)$$



[1] Ying, Rex, et al. "Gnnexplainer: Generating explanations for graph neural networks." NeurIPS, 2019.
[2] Luo, Dongsheng, et al. "Parameterized explainer for graph neural network." NeurIPS, 2020.
[3] Yuan, Hao, et al. "Xgnn: Towards model-level explanations of graph neural networks." SIGKDD 2020.

Notations

We consider a graph classification model $f(\cdot)$, such that y is the predicted class for input graph G.

We want to explain the outcome y by extracting a *connected* subgraph.

$$G^* = \arg \max_{|G_i| \leq N_{min}} Score(f, G, G_i)$$

$|G_i| \leq N_{min}$ is a connected subgraph with no more than N_{min} number of nodes;

$Score(\cdot,\cdot,\cdot)$ is a scoring function evaluating the importance of G_i given f and G.

Solve by searching.

1. Brute-force is intractable and thus employ Monte Carlo Tree Search (MCTS).
2. Use Shapley value as scoring function.

Exploring subgraph by MCTS

Build a search tree:
Root is the input graph;

Each node N_i is a connected graph;

Each edge is a node-pruning action a:

[Remove a node, all connected edges are removed.

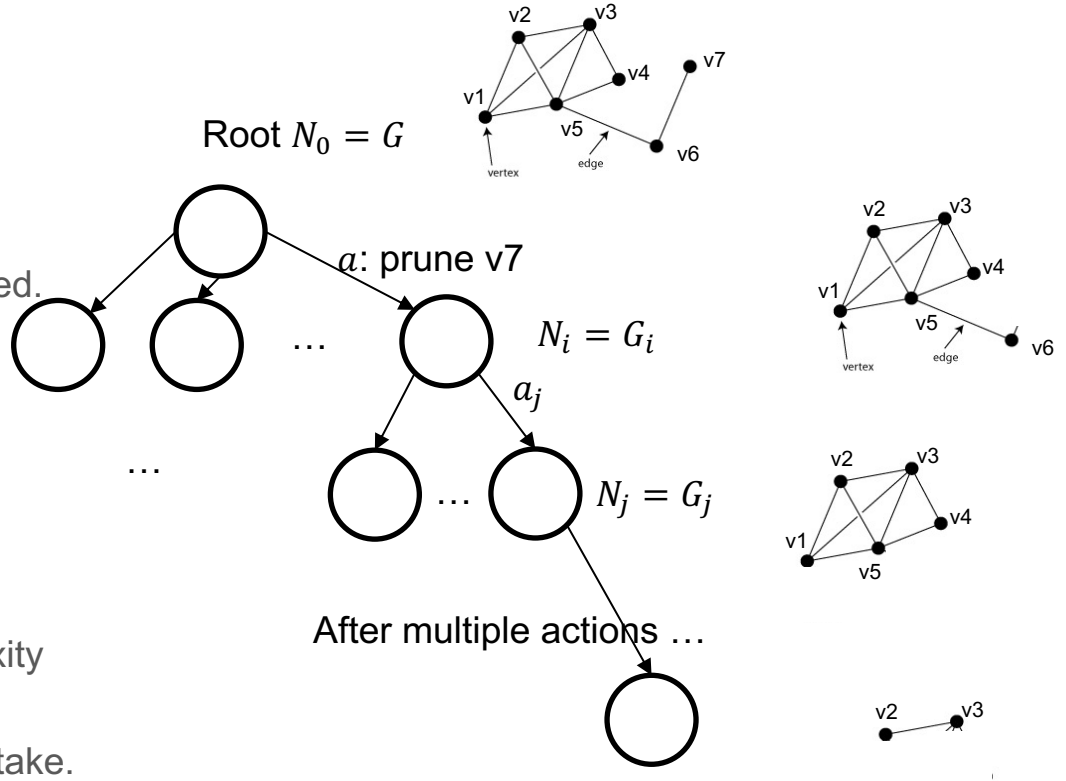
If there are multiple disconnected subgraphs,

retain the largest one.]

Pair (N_i, a_j): G_j is obtained by pruning a_j from N_i.

If we search all possible leaf nodes, time complexity is exponential (combination problem);

MCTS provides the clues about which actions to take.



Exploring subgraph by MCTS

After building the search tree, in MCTS:

For each pair (N_i, a_j), we record four variables:

$\mathrm{C}(N_i, a_j)$: the number of counts selecting a_j from N_i.
$\mathrm{W}(N_i, a_j)$: the total reward for all (N_i, a_j).

$\mathrm{Q}(N_i, a_j) = W(N_i, a_j)/\mathrm{C}(N_i, a_j)$: the average reward for (N_i, a_j).

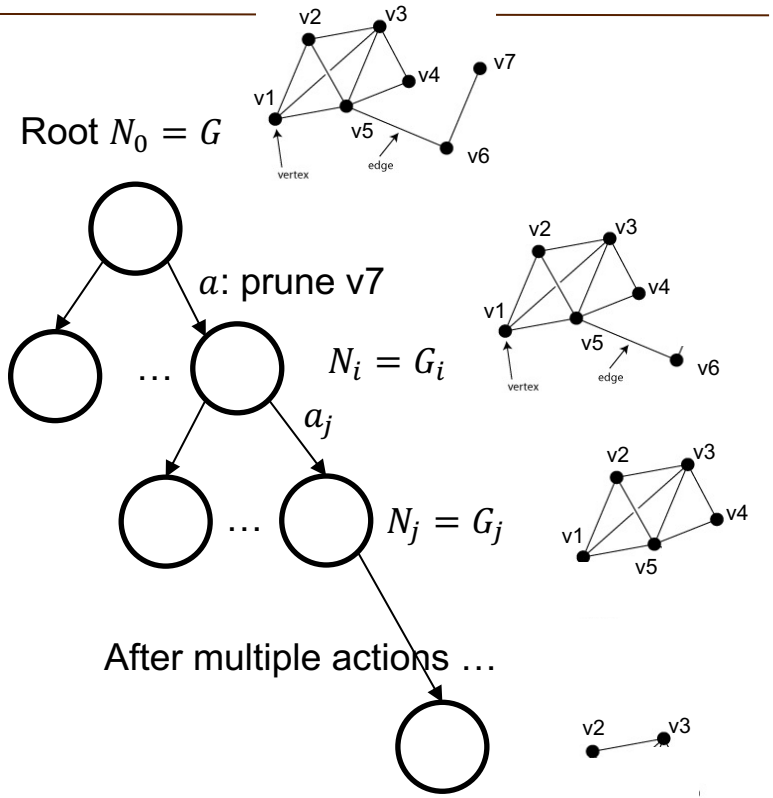
$\mathrm{R}(N_i, a_j) = Score(f, G, G_j)$: immediate reward measuring the importance of G_j.

The criteria for action selection is:

$$a^* = \arg\max_{a_j} Q(N_i, a_j) + U(N_i, a_j)$$

$$U(N_i, a_j) = \lambda R(N_i, a_j) \frac{\sqrt{\sum_k C(N_i, a_k)}}{1 + C(N_i, a_j)}$$

Then update four variables in the path.



The score function – Shapley Value

$Score(\cdot,\cdot,\cdot)$ is used in explanation quality evaluation and the MCTS rewards.

If we use the predicted scores from the trained GNN f for subgraph G_i, "it cannot capture the interactions between different graph structures, thus affecting the explanation results."

We use Shapley values: the GNN predication is the game gain, and graph structures are players.

The score function – Shapley Value

To study SV of a subgraph G_i with $V = \{v_1, \dots, v_k\}$ from a graph G with $V = \{v_1, \dots, v_i, \dots, v_m\}$.

The set of players $P = \{G_i, v_{k+1}, \dots, v_m\}$, and SV of the player G_i is:

$$\phi(G_i) = \sum_{S \subseteq P \setminus \{G_i\}} \frac{|S|!\,(|P| - |S| - 1)!}{|P|!} m(S, G_i)$$

$$m(S, G_i) = f(S \cup \{G_i\}) - f(S)$$

S is the possible coalition set of players, m is the marginalized contribution.

Time consuming! It enumerates all possible coalitions.

-> Only consider L-hop neighborhood of G_i (GNN f contains L layers)

Replace P by $P' = \{G_i, v_{k+1}, \dots v_r\}$ (within L-hop neighborhood)

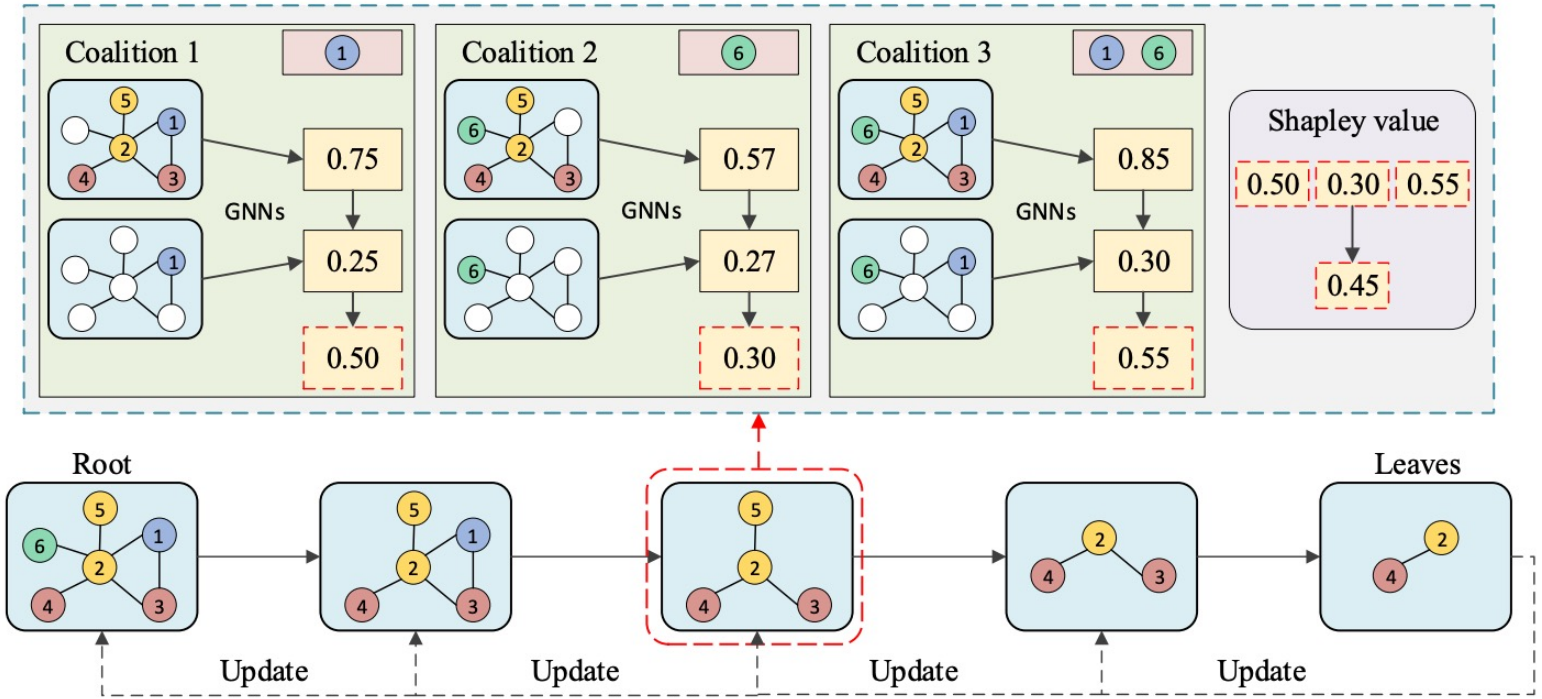
-> Monte-Carlo sampling

Sample T coalition sets S_i, and the averaged contribution score is regarded as the approximation:

$$\phi(G_i) = \frac{1}{T} \sum_{t=1}^{T} m(S_i, G_i)$$

Overview of SubgarphX

Upper row: compute SV by MC sampling; lower row: explore subgraph in a search tree.



Experiments

Datasets: [molecular: MUTAG, BBBP] [Sentiment: Graph-SST2] [synthetic: BA-2Motifs, BA-shape]

Target model: GCNs, GATs, GINs.

Baselines: GNNExplainer, PGExplanier, MCTS_GNN

MCTS_GNN uses MCTS to explore subgraphs but employs the GNN predictions of these subgraphs as the scoring function.

Qualitative experiments (no group truth)

Quantitative metrics: Fidelity, sparsity and efficiency.

Fidelity: it removes the important structure from the input graphs, and computes the difference between predictions.

Sparsity: the fraction of structures being identified as important.

Efficiency: running time.

Experiments

Qualitative experiments:

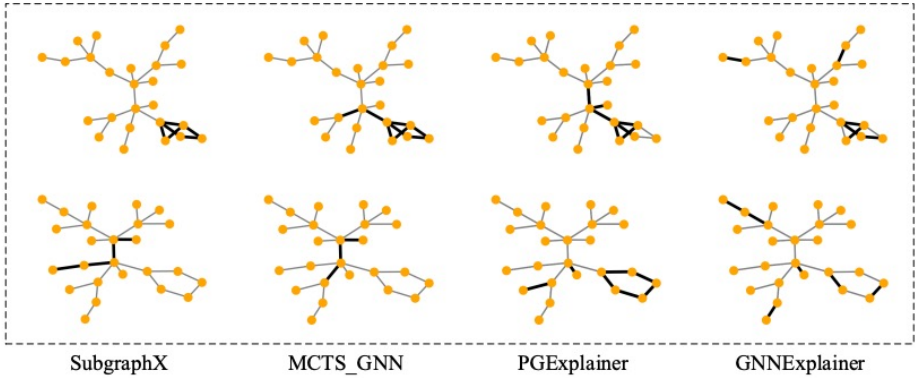


Figure 3. Explanation results on the BA-2Motifs dataset with a GCN graph classifier. The first row shows explanations for a correct prediction and the second row reports the results for an incorrect prediction.

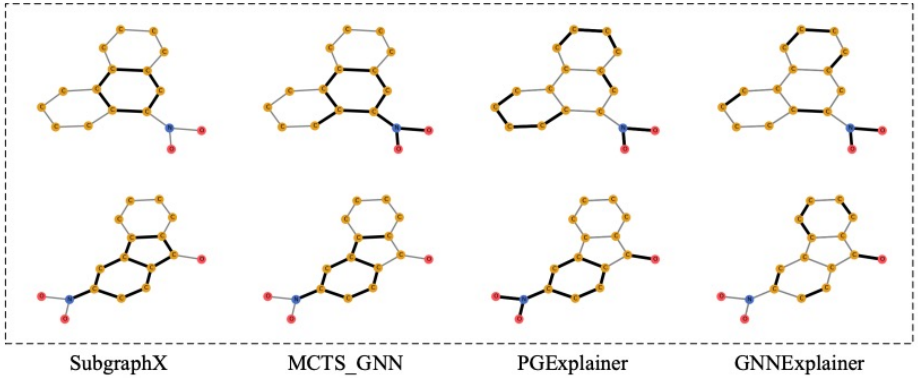
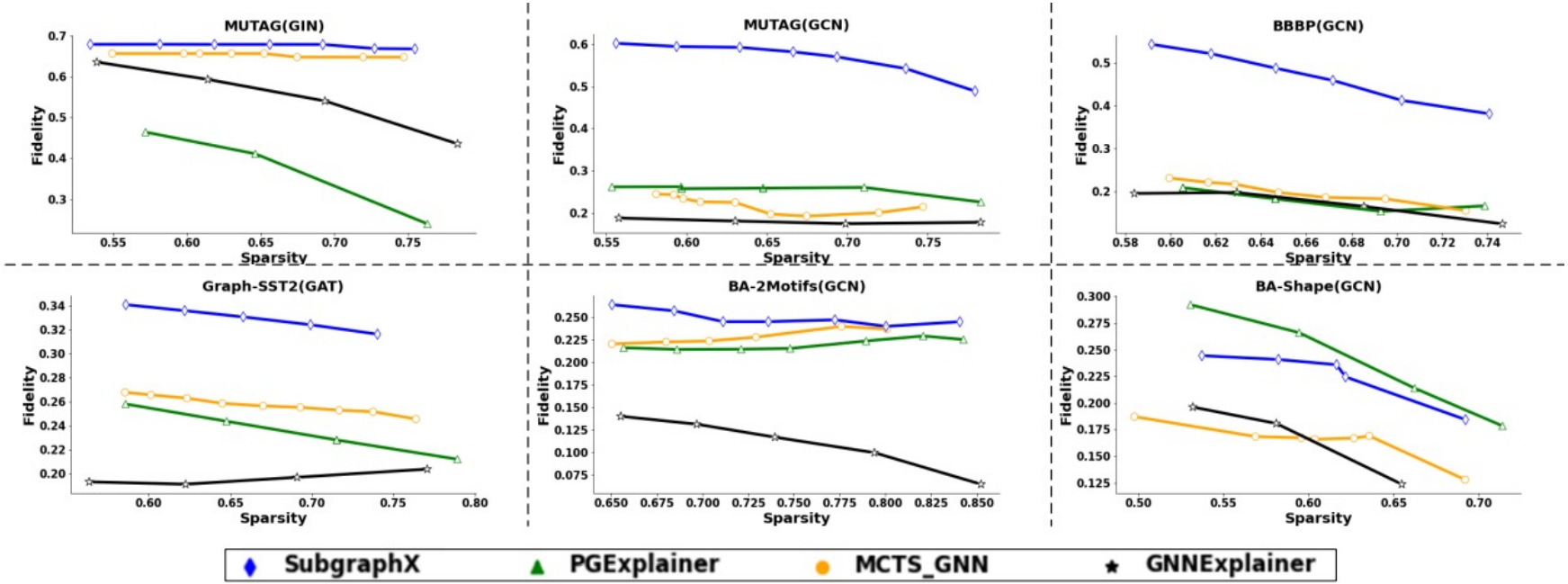


Figure 4. Explanation results on the MUTAG dataset with a GIN graph classifier. We show the explanations for two correct predictions. Here Carbon, Oxygen, and Nitrogen are shown in yellow, red, and blue, respectively.

Experiments

Quantitative experiments: (Fidelity and Sparsity)



Experiments

Quantitative experiments: (Efficiency) [For 50 graphs with an average of ~25 nodes in BBBP]

Table 2. Efficiency studies of different methods.

Method	MCTS*	MCTS†	SubgraphX	GNNExplainer	PGExplainer
TIME	>10 hours	865.4 ± 1.6s	77.8 ± 3.8s	16.2 ± 0.2s	0.02s (Training 362s)
FIDELITY	N/A	0.53	0.55	0.19	0.18

MCTS* doesn't use Monte Carlo sampling: $\phi(G_i) = \sum_{S \subseteq P' \setminus \{G_i\}} \frac{|S|!(|P|-|S|-1)!}{|P|!} m(S, G_i)$

MCTS+ uses MC sampling but doesn't use approximation P'.

Experiments

Pruning strategy:

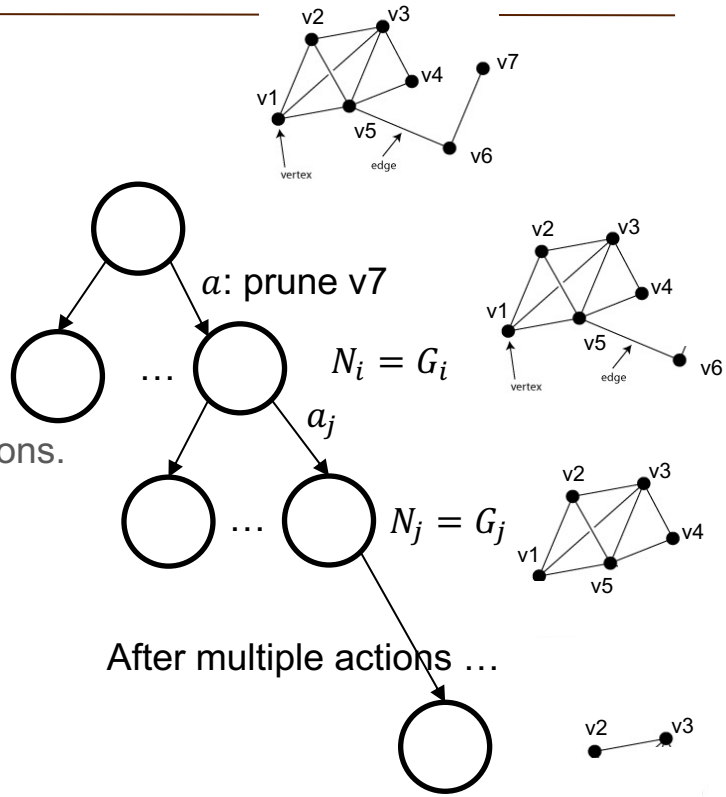
1. Low2High:

Arranges the nodes based on node degrees from low to high.

Only consider to prune top-k lowest degree nodes.

2. High2Low is similar, but prunes top-k highest degree nodes.

High2Low should be more efficient but may ignore optimal solutions.



Method	Time	Fidelity
LOW2HIGH	107.24s	0.66149
HIGH2LOW	21.52s	0.61046

Thank you

LEHIGH UNIVERSITY