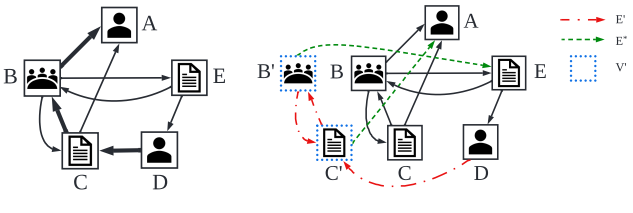
Towards Fine-grained Explainability for Heterogeneous Graph Neural Network

Tong Li[1], Jiale Deng[1], Yanyan Shen[1*], Luyu QIU[2], Yongxiang Huang[2], Caleb Chen Cao[2]

[1] Shanghai Jiao Tong University, [2] Huawei Research Hong Kong
{2017lt, jialedeng, shenyy}@sjtu.edu.cn, {qiuluyu, huang.yongxiang2, caleb.cao}@huawei.com



(a) (b)

Figure 1: (a) Illustration of the fine-grained explanation ($v=D$, $P=\langle D,C,B,A\rangle$) for the prediction of A on graph G. (b) The rewired graph G_R^P where B', C' are proxies.

A. Proof of Rewiring Algorithm Correctness

We first formally define the set of walks that end with the target node v_t.

Definition 1 *Given a walk W on G, $W=\langle v_1,\cdots,v_L\rangle$. $W\in\mathcal{W}_G^{v_t}$ iff $v_L=v_t$ and $v_i\neq v_t, \forall 1\leq i<L$.*

Note that the set only contains walks that reach v_t once. The reason behind such a definition is that we assume a walk contributes to the prediction when it first reaches v_t, while the part that walks out and returns to v_t is regarded as the influence of v_t to itself. In xPath , we do not consider fine-grained explanations where the cause node is the target node itself (the simple path of length 0), because humans can easily understand how the target node influences the prediction of itself without the aid of other graph objects.

This section is to give a detailed proof of the following theorem in our paper.

Theorem 1 *For any fine-grained explanation $X_G(v_t)=(v,P)$ for the prediction $M_G(v_t)$, Algorithm 1 produces a rewired graph G_R^P that satisfies: (i) $\mathcal{W}_G^P\cap\mathcal{W}_{G_R^P}^{v_t}=\emptyset$; and (ii) $\mathcal{W}_G^P\cup\mathcal{W}_{G_R^P}^{v_t}=\mathcal{W}_G^{v_t}$.*

For ease of representation, we first define some node and edge sets notifications.

- $V'=\bigcup_{i=1}^{L}\{\text{proxy}(v_i)\}$, is the set of proxies we created for G_R^P.
- $E'=\bigcup_{i=1}^{L}\{\mathsf{InE}(\text{proxy}(v_i))\}$ is the set of in-edges of proxies on G_R^P. E' are exactly the edges we add to G_R^P by line 5 of Algorithm 1.
- $E^*=\bigcup_{i=1}^{L}\{\mathsf{OutE}(\text{proxy}(v_i))\}-E'$ is the set of edge on G_R^P that is an out-edge of some proxy but not an in-edge of any proxy. E^* is exactly the edges we add to G_R^P in line 10 of Algorithm 1.

Denote by V the original node set of G, the node set of G_R^P is $V\cup V'$. Denote by E the original edge set of G, the edge set of G_R^P is $E\cup E'\cup E^*\backslash\{\langle v,v_1\rangle\}$. We show an example in Figure 1b. Moreover, we denote the associated edge set of a simple path P as $\mathsf{PE}(P)$. For each edge e in G, $e\in\mathsf{PE}(P)$, if e is a self-loop on a node in the path, **or** either itself or its reverse is in the path. If a walk is associated with P with a valid suffix satisfies Definition 5, each edge in the suffix is in $\mathsf{PE}(P)$. For example on G shown in Figure 1a, $\mathsf{PE}(P=\langle D,C,B,A\rangle)=\{\langle D,C\rangle,\langle C,B\rangle,\langle B,A\rangle,\langle B,C\rangle\}$.

Let a simple path P=$\langle v(v_0),v_1,\cdots,v_L,v_t(v_{L+1})\rangle$, $L\geq 0$. We prove Theorem 1 by the following lemmas.

Lemma 1 $\mathcal{W}_G^P\cap\mathcal{W}_{G_R^P}^{v_t}=\emptyset$.

Proof. We prove Lemma 1 by showing that for a walk $W_1\in\mathcal{W}_G^P$, it is impossible to find a walk $W_2\in\mathcal{W}_{G_R^P}$ such that $W_1=W_2$. Recall that every node and edge in G has a distinct type. In G_R^P, we have the following facts about equivalent graph objects with those in G: (i) for a node in G but not in P, the only node sharing the same type and feature vector with it is itself. (ii) for a node in G and in P, the node sharing the same type and feature vector with v_i except itself is $\text{proxy}(v_i)\in V'$. (iii) for an edge $e\in\mathsf{PE}(P)$, the only edge sharing the same type and feature vector with e except itself is a corresponding edge $e'\in E'$.

We denote by $W_{1,s}$ the shortest suffix of W_1 that satisfies the three conditions in Definition 5. If $W_1=W_2$, W_2 mush have a suffix $W_{2,s}=W_{1,s}$. We now check the walking process of $W_{1,s}$ and $W_{2,s}$ step by step. Note that $W_{1,s}$ is on G, while $W_{2,s}$ is on G_R^P. As $W_{1,s}$ is the shortest valid suffix, its first step is $\langle v,v_1\rangle$. To find an equivalent edge, the first step of $W_{2,s}$ must be $\langle v,\text{proxy}(v_1)\rangle$ because we have remove the other possible edge $\langle v,v_1\rangle$ from the graph. After the first step, every edge of $W_{1,s}$ is in $\mathsf{PE}(P)$ and $W_{2,s}$ can only choose an edge from E' to keep the step equivalent. Note that after the first step, $W_{2,s}$ is on V' and by walking

[*]corresponding author.

along the edges in E' it can never return to V. Because there is only one edge in E' that connects a node in V' to V, which is $\langle \mathrm{proxy}(v_1), v \rangle$ (if exists). But we choose $W_{1,s}$ to be the shortest suffix, which can never contain $\langle v_1, v \rangle$. So, $W_{2,s}$ can never return to V, implying it cannot reach v. Therefore, we fail to construct a walk W_2 for W_1.

Lemma 2 $(\mathcal{W}_G^{v_t} - \mathcal{W}_G^P) \subset \mathcal{W}_{G_R^P}^{v_t}$.

Proof. We prove Lemma 2 by showing that for any walk $W_1 \in (\mathcal{W}_G^{v_t} - \mathcal{W}_G^P)$, we can find a walk $W_2 \in \mathcal{W}_{G_R^P}$ such that $W_1 = W_2$.

Let $u \in V$ be the first node of W_1. On G_R^P, we start with $u \in V$ and show the process to find W_2 step by step. When we are on V, for every step r of W_1, we first try to find an equivalent edge in E for W_2. We can always find such an edge except for $r = \langle v, v_1 \rangle$. If we encounter $\langle v, v_1 \rangle$, we choose $\langle v, \mathrm{proxy}(v_1) \rangle \in E'$ and begin to walk on V'. When we are on V', for every step r of W_1, it is impossible that $r = \langle v_L, v_t \rangle$, otherwise W_1 will have a suffix that satisfies the three conditions in Definition 5, leading to $W_1 \notin (\mathcal{W}_G^{v_t} - \mathcal{W}_G^P)$. Therefore, (i) if $r \in \mathsf{PE}(P)$ and $r \neq \langle v_L, v_t \rangle$, we can find an equivalent edge in E' and keep walking on V', (ii) if $r \notin \mathsf{PE}(P)$, we can find an equivalent edge in E^* and return to V. With above strategy, we can find $W_2 = W_1$ step by step until W_1 reaches v_t.

Lemma 3 $\mathcal{W}_{G_R^P}^{v_t} \subset \mathcal{W}_G^{v_t}$

Proof. Note that equivalent walks are defined only by the types and feature vectors of ordered graph objects in the walks, but do not distinguish two graph objects with the same type and feature vector. We attempt to merge every $\mathrm{proxy}(v_i), i \in [1, L]$ in G_R^P into v_i as they share the same type and feature vector. Specifically, for every in-edge $\langle u, \mathrm{proxy}(v_i) \rangle \in \mathsf{InE}(\mathrm{proxy}(v_i))$, we connect u to v_i with an edge of the same type and feature vector. For every out-edge $\langle \mathrm{proxy}(v_i), u \rangle \in \mathsf{OutE}(\mathrm{proxy}(v_i))$, we connect v_i to u with an edge of the same type and feature vector. Then, we remove the proxies and their connecting edges. Denote the resultant graph as G_M^P, whose node set is V and edge set is E^M. Note that the in-edges and out-edges are in $E \cup E^*$ and they have corresponding equivalent edges in E. So, $E^M \subset E$ and $\mathcal{W}_{G_M^P}^{v_t} \subset \mathcal{W}_G^{v_t}$. Also, it is obvious that every walk is kept during the merging process, which implies $\mathcal{W}_{G_R^P}^{v_t} \subset \mathcal{W}_{G_M^P}^{v_t}$. Therefore, $\mathcal{W}_{G_R^P}^{v_t} \subset \mathcal{W}_G^{v_t}$.

B. Evaluation of Greedy Search

We investigate the improvement brought by the greedy search algorithm and compare it with random sampling. To randomly sample a simple path that ends with v_t, we perform a random walk on the inverted graph of G. On the inverted heterogeneous graph, we start a random walk from v_t and obey a certain metapath type while randomly selecting a successor that has never been visited yet. We set the sample number $m = 10$ for each metapath type. We report the fidelity scores of top-K fine-grained explanations in Figure 2 and the average time taken to generate explanations for each sample in Figure 2. Compared with random sampling, greedy search shows an average 27% diminution on F^{acc}, 54% on F^{prob}, and 56% on time cost, respectively. This demonstrates that with greedy search, xPath is able to discover higher-quality fine-grained explanations more efficiently.

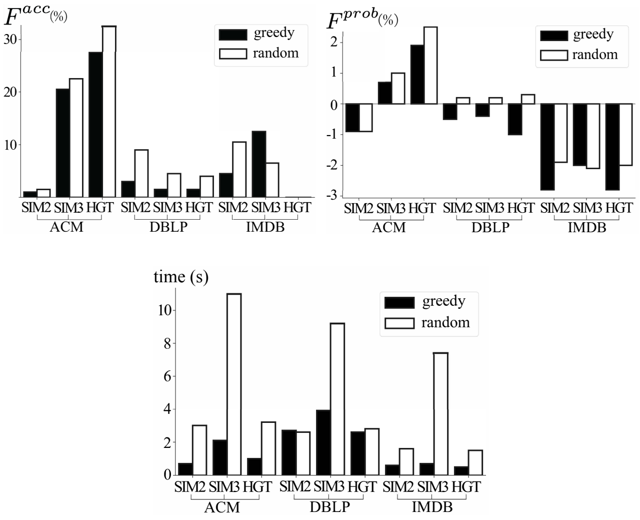


Figure 2: Comparison of greedy search algorithm with random sampling.