Generating Skyline Explanations for Graph Neural Networks

Dazhuo Qiu
Aalborg University
Denmark
dazhuoq@cs.aau.dk

Haolai Che
Case Western Reserve
University, USA
hxc859@case.edu

Arijit Khan
Aalborg University
Denmark
arijitk@cs.aau.dk

Yinghui Wu
Case Western Reserve
University, USA
yxw1650@case.edu

ABSTRACT

This paper proposes a novel approach to generate subgraph explanations for graph neural networks (GNNs) that simultaneously optimize multiple measures for explainability. Existing GNN explanation methods often compute subgraphs (called "explanatory subgraphs") that optimize a pre-defined, single explainability measure, such as fidelity or conciseness. This can lead to biased explanations that cannot provide a comprehensive explanation to clarify the output of GNN models. We introduce *skyline explanation*, a GNN explanation paradigm that aims to identify k explanatory subgraphs by simultaneously optimizing multiple explainability measures. (1) We formulate skyline explanation generation as a multi-objective optimization problem, and pursue explanations that approximate a skyline set of explanatory subgraphs. We show the hardness for skyline explanation generation. (2) We design efficient algorithms with an onion-peeling approach that strategically removes edges from neighbors of nodes of interests, and incrementally improves explanations as it explores an interpretation domain, with provable quality guarantees. (3) We further develop an algorithm to diversify explanations to provide more comprehensive perspectives. Using real-world graphs, we empirically verify the effectiveness, efficiency, and scalability of our algorithms.

1 INTRODUCTION

Graph neural networks (GNNs) have demonstrated promising performances in graph analytical tasks such as classification. Given a graph G (a network representation of a real-world dataset), a GNN $\mathcal{M}$ aims to learn the node representations of G that can be converted to proper results for targeted downstream tasks, e.g., node classification, link prediction, or regression analysis. For example, a GNN-based node classification assigns a class label to a set of test nodes in G, where the label of each test node v (the "output" of $\mathcal{M}$ at node v, denoted as $\mathcal{M}(v, G)$) is determined by the node representation learned by the GNN $\mathcal{M}$. GNNs have been applied for node classification in biochemistry, social and financial networks [14, 55, 56, 61], among other graph analytical tasks.

Despite their promising performance, it remains desirable yet nontrivial to explain the output of GNNs to help users understand their behavior [62]. Several GNN explainers are proposed to generate subgraphs (called "explanatory subgraphs") that are "responsible" to clarify the output of $\mathcal{M}$ over G [34, 35, 52, 60, 66]. For example, given $\mathcal{M}$ as a GNN-based node classifier and a test node v in the graph G, a GNN explainer computes an explanatory subgraph G_ζ of G that can best clarify the node label $\mathcal{M}(v, G)$. This is often addressed by solving an optimization problem that discovers a subgraph G_ζ subject to a pre-defined metric, which quantifies the explainability of G_ζ for the output $\mathcal{M}(v, G)$ for a test node v.

Prior work typically pre-assumes and optimizes a single metric of interest, such as fidelity, sparsity, or stability to generate explanatory subgraphs with high explainability [62]. Such a metric assesses explanations from a pre-defined, one-sided perspective of explainability (as summarized in Table 2, § 2). For example, a subgraph G_ζ of a graph G is a "*factual*" explanation for $\mathcal{M}$ over G, if it preserves the output of $\mathcal{M}$ (hence is "faithful" to the output of $\mathcal{M}$) [33, 35, 52, 60]. G_ζ is a "*counterfactual*" explanation, if removing the edges of G_ζ from G leads to a change of the output of $\mathcal{M}$ on the remaining graph (denoted as $G \setminus G_\zeta$) [33, 34, 52]. Other metrics include fidelity$^-$ [33, 34, 52] (resp. fidelity$^+$ [35, 63]), which quantifies the explainability of G_ζ in terms of the closeness between the task-specific output of $\mathcal{M}$, such as the probability of label assignments, over G_ζ (resp. $G \setminus G_\zeta$) and their original counterpart G, and "*conciseness* (*sparsity*)" that favors small explanatory subgraphs.

Nevertheless, GNN explainers that optimize a single metric may lead to biased and less comprehensive explanations. For example, an explanation that achieves high fidelity may typically "compromise" in conciseness, due to the need of including more nodes and edges from the original graphs to be "faithful" to the GNN output. Consider the following example.

Example 1: In Bitcoin blockchain transactions, money launderers employ various techniques to conceal illicit cryptocurrency activities and evade detection by law enforcement agencies and AI-based monitoring systems [39, 43]. Figure 1 illustrates an input graph G that includes account IP addresses and Bitcoin transactions among the accounts. Each IP address is associated with transaction-related features: the number of blockchain transactions (Txs), the number of transacted Bitcoins (BTC), and the amount of fees in Bitcoins (Fee). The task is to detect illicit IP addresses with a GNN-based node classifier. A GNN classifier has correctly detected v_t as "*illicit*".

A law enforcement agency wants to understand why the GNN asserts v_t as an illicit account address. They may ask an "explanatory query" that requests to generate explanations ("*which fraction of the graph G are responsible for the* GNN*'s decision of assigning the label "illicit" to the account* v_t*?*"), and further ground this output with real-world evidences (*e.g.,* by referring to known real-world money laundering scenarios [5, 12]). Therefore, it is desirable to generate explanatory subgraphs as intuitive and natural showcases of money laundering scenarios. For example, "*Spindle*" [12] suggests that perpetrators generate multiple shadow addresses to transfer small amounts of assets along lengthy paths to a specific destination; and *Peel Chain* [5] launders large amounts of cryptocurrency through sequences of small transactions, where minor portions are 'peeled' from the original address and sent for conversion into fiat currency to minimize the risk of being detected.

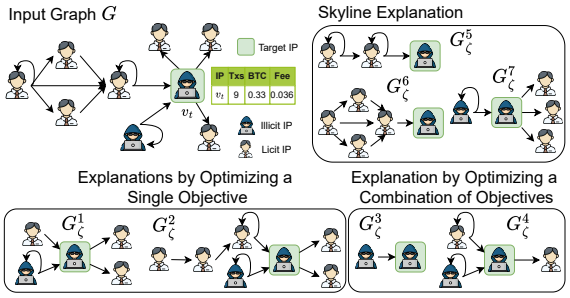


Figure 1: A Bitcoin transaction network with a target IP address (test node v_t) that has a label "*illicit*" to be explained [19].

Consider the following explanatory subgraphs, generated by representative GNN explainers: G_ζ^1 is a factual explanatory subgraph generated by the explainer in [60] (a "factual explainer"); and G_ζ^2 is a counterfactual explanatory subgraph generated by a "counterfactual explainer" [34]. Compare G_ζ^1 and G_ζ^2. (1) G_ζ^1 includes a subgraph induced by v_t and most of its neighbors, which is indeed a critical fraction that can preserve the output of a GNN's output; nevertheless, it misses important nodes that have a higher influence on GNN's decision making, but are not in v_t's direct neighborhood. (2) Such nodes can be captured by a counterfactual explanatory subgraph, as depicted in G_ζ^2. Although G_ζ^2 can capture more nodes with high influence beyond v_t's neighbors, it is enforced to include a larger fraction of G to ensure that the removal of its edges incurs great enough impact to change the output of the GNN classifier, hence sacrificing "conciseness", conflicting for users who favor quick generation of small evidences that are "faithful" to original GNN output [33]. Choosing either alone can be biased to a "one-sided" explanation for the "illicit" IP address. □

Can we generate explanations that simultaneously optimize multiple explainability metrics? One solution is to compute subgraphs that optimize a linear combination of all metrics. This aggregates multiple criteria into a single objective function using a set of weights. However, such weighted sum methods may lead to a single, marginally optimal answer across all criteria, overlooking other high-quality and diverse solutions, hence an overkill [18, 36].

Example 2: Consider another two explanatory subgraphs: G_ζ^3 is the explanation generated by an explainer that optimizes both conciseness and factual measures [35]; G_ζ^4 is the explanation generated by an explainer that linearly combines factual and counterfactual measures into a single, bi-criteria objective function to be optimized [52]. Such simple combinations enforce explainers to optimize potentially "conflicting" measures, or highly correlated ones, either may result in lower-quality solutions that are sensitive to data bias. For example, as factual measures and conciseness may both encourage smaller and faithful explanations, explanatory subgraphs obtained by [35], such as G_ζ^3, turn out to be relatively much smaller and less informative, hardly providing sufficient real-world evidence that can be grounded by money laundering behaviors. On the other hand, explanatory subgraphs from [52] such as G_ζ^4 may easily be a one-sided explanation of either factual or counterfactual. Indeed, we find that such explanations capture only one type of money laundering scenario at best in most cases (*e.g., Spindle* [12]). □

These examples illustrate the need to generate explanations via a multi-objective optimization paradigm, leading to more comprehensive and balanced outcomes. Skyline query processing has been extensively studied [13, 42]. Analogizing well-established skyline queries [6, 9, 29, 32, 40] which compute Pareto sets (also referred to as "skylines") that are data points not dominated by each other across a set of quality measures, we advocate approaching GNN explanation by generating explanatory subgraphs that are Poreto sets over multiple user-defined explanatory measures. Skylines (or "Pareto set") generally offer better solutions against the aforementioned alternatives [18, 50]. We refer to such explanations "*skyline explanations*", and illustrate an example below.

Example 3: Consider a set of subgraphs G_ζ^5, G_ζ^6, and G_ζ^7. These explanatory subgraphs are selected as a Pareto-optimal set across three explanatory measures: fidelity+, fidelity-, and conciseness. Each subgraph is high-quality, diverse, and non-dominated in at least one measure that is higher than others in the set. This result provides a more comprehensive and intuitive interpretation to explain "why" v_t is identified as "illicit" by GNN-based classification. Indeed, G_ζ^5, G_ζ^6, and G_ζ^7 capture different money laundering scenarios: *Peel Chain* [5], *Spindle* [12], and a combination, respectively. Therefore, our identified skyline explanations G_ζ^5, G_ζ^6, and G_ζ^7 better support law enforcement agencies by distinguishing various money laundering evidences that v_t is involved. □

We advocate to develop a GNN explainer that can efficiently generate skyline explanations for large-scale GNN-based analysis. Such an explainer should: (1) generate skyline explanations for designated output of interests and any user-defined set of explanatory measures; and (2) generate a diversified set of skyline explanations upon request, and (3) ensure desirable guarantees in terms of Pareto-optimality. The need for skyline explanations are evident for trustworthy and multifaceted analysis and decision making, as observed in *e.g.,* drug repurposing [44], cybersecurity analysis [24], fraud detection [39], social recommendation [20], among others, where GNN output should be clarified from multiple, comprehensive aspects, rather than one-sided, biased perspectives.

Contribution. This paper formulates and investigates a novel problem of generating skyline explanations in terms of explanatory subgraphs, by simultaneously optimizing multiple user-specified explanatory measures. We summarize our contribution as follows. (1) *A formulation of Skyline Explanation Generation problem.* We introduce a class of skyline explanatory query (SXQ) to express the configuration for generating skyline explanations. An SXQ takes as input a graph G, a GNN $\mathcal{M}$, nodes of interests V_T, and a set of explainability measures Φ, and requests a skyline explanation (a set of explanatory subgraphs) that clarifies the output of $\mathcal{M}$ over each node in V_T, that simultaneously optimizes the measures in Φ.

The evaluation problem of SXQ is to generate a skyline explanation (a set of k explanatory subgraphs) as a comprehensive explanation of the output of $\mathcal{M}$ for V_T. We approach the problem with multi-objective optimization, based on a subgraph dominance

relation over explainability measures Φ. We verify the hardness of both the decision and optimization version of the problem.

(2) We introduce efficient algorithms to process SXQs in terms of Pareto optimality. The algorithm adopts an "onion-peeling" strategy to iteratively reduce edges at each hop of the targeted nodes, and validates a bounded number of generated explanations in a discretized coordination system to incrementally improve the quality of the answer. We show that this process ensures a $(1 + \epsilon)$ approximation of Pareto-optimal set. We also present an algorithm to diversify the answer set for SXQs.

(3) Using real-world graphs and benchmark tasks from various domains, we show qualitative and quantitative analysis to verify the effectiveness and scalability of our GNN explainers. We visualize skyline explanations with corresponding distribution, to showcase applications of our novel problems and algorithms. Our approach efficiently generates a set of skyline explanations for nodes of interest, even on large-scale graphs. For example, we outperform CF^2 [52] in the Integrated Preference Function score by 2.8× on Cora dataset; for OGBN_arxiv dataset with million-scale edges, we outperform the fastest baseline GNNExplainer [60] by 1.4×.

Related Work. We categorize related work into the following.

Graph Neural Networks. GNNs have demonstrated themselves as powerful tools in performing various graph learning tasks, including, but not limited to node classification, link prediction [64], and graph classification. [60] Recent studies proposed multiple variants of the GNNs, such as graph convolution networks (GCNs) [28], graph attention networks (GATs) [54], Graph Isomorphism Networks (GINs) [57]. These methods generally follow an information aggregation scheme where features of a target node are obtained by aggregating and combining the features of its neighboring nodes.

Explanation of GNNs. Several GNN explanation approaches have been studied. (1) *Learning-based* methods aim to learn substructures of underlying graphs that contribute to the output of a GNN. GNNExplainer [60] identifies subgraphs with node features that maximize the influence on the prediction, by learning continuous soft masks for both adjacency matrix and feature matrix. CF-GNNExplainer [34] learns the counterfactual subgraphs, which lead to significant changes of the output if removed from the graphs. PG-Explainer [35] parameterizes the learning process of mask matrix using a multi-layer perceptron. GraphMask [47] learns to mask the edges through each layer of GNNs that leads to the most sensitive changes to their output. (2) Learning-based GNN explainers often require prior knowledge of model parameters and incur considerable learning overhead for large graphs. Other approaches perform post-processing to directly generate explanatory subgraphs that optimize a pre-assumed explanation criteria. SubgraphX [63] utilizes Monte-Carlo tree search to compute subgraphs that optimize a game-theory-inspired Shapley value. GStarX [65] follows a similar approach, yet aims to optimize "HN values", a topology-aware variant of Shapley value. These methods typically focus on optimizing a single, pre-defined criterion, and cannot provide a configurable mechanism for users to customize the generation of explanations.

Closer to our setting are GNN explainers that aim to optimize more than one explainability criterion. CF^2 [52] leverages both factual and counterfactual reasoning to formulate an objective that is a linear function of weighted factual and counterfactual measures. It learns feature masks and edge masks aiming to produce explanations that optimize this objective function. RoboGExp [45] generates subgraph explanations that are factual, counterfactual, and meanwhile robust, *i.e.,* explanatory subgraphs remain invariant structures under a bounded number of edge modifications. GEAR [66] learns GNN explainers by adjusting the gradients of multiple objectives geometrically during optimization. GEAR handles gradient conflicts globally by selecting a dominant gradient based on user-desired objectives (such as fidelity) and adjusting conflicting gradients to lie within a controlled angle threshold. MOExp [33] introduces a bi-objective optimization algorithm to find Pareto optimal explanations that strike a balance between "simulatability" (factual) and "counterfactual relevance". It proposes a zero-order search algorithm that optimizes without accessing the target GNN model's architecture or parameters, making it universally applicable. Despite these methods generating explanations that can address multiple criteria, the overall goal remains pre-defined – they do not provide *configurable* manner to allow user-defined preferences as needed. In addition, MOExp does not control the size of explanations, which may result in a large set of subgraphs that are hard to inspect.

Skyline queries. Multi-objective search and skyline queries have been extensively studied [6, 15, 16]. These approaches compute Pareto optimal sets [13, 26] or their approximate variants [30, 41] over data points and a set of optimization criteria. Notable strategies include [26] that transform multiple objectives into a single-objective counterpart. Constraint-based methods such as [13] initialize a set of anchor points that optimize each single measure, and bisect the straight lines between pairs of anchor points with a fixed vertical separation distance. This transforms bi-objective optimization into a series of single-objective counterparts. Solving each derives an approximation of the Pareto frontier. ϵ-Pareto set [30, 41] has been widely recognized as a desirable approximation for the Pareto optimal set. While these algorithms cannot be directly applied to answer SXQ, we introduce effective multi-objective optimization algorithms to generate explanatory subgraphs with provable quality guarantees in terms of ϵ-Pareto approximation.

2 GRAPHS AND GNN EXPLANATION

Graphs. A directed graph $G = (V, E)$ has a set of nodes V and a set of edges $E \subseteq V \times V$. Each node v carries a tuple $T(v)$ of attributes and their values. The size of G, denoted as $|G|$, refers to the total number of its edges, *i.e.,* $|G| = |E|$. Given a node v in G, (1) the L-hop neighbors of v, denoted as $N^L(v)$, refers to the set of nodes in the L-hop of v in G. (2) The *L-hop neighbor subgraph*, denoted as $G^L(v)$, refers to the subgraph of G induced by $N(v)$. (3) The L-hop neighbor subgraph of a set of nodes $V_s \subseteq V$, denoted as $G^L(V_s)$, refers to the subgraph induced by the node set $\bigcup_{v \in V_s} N^L(v)$.

Graph Neural Networks. GNNs [28] comprise a well-established family of deep learning models tailored for analyzing graph-structured data. GNNs generally employ a multi-layer message-passing scheme as shown in Equation 1.

$$\mathbf{H}^{(l+1)} = \sigma(\widetilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \quad (1)$$

Table 1: Summary of notations

Notation	Description
$G = (V, E)$	a graph G with node set V and edge set E
X, $\widetilde{\mathbf{A}}$	X: feature matrix, $\widetilde{\mathbf{A}}$: normalized adjacency matrix
$\mathcal{M}, L$	a GNN model with number of layers L
$H^{(l)}(v, G)$; $H(v, G)$	embedding of node v in G at layer l ($l \in [1, L]$)
$\mathcal{M}(G)$, $\mathcal{M}(v, G))$	task-specific output of $\mathcal{M}$ over G and over $v \in V$
V_T	a set of (test) nodes of interests
s; G_s	a state s; a candidate G_s
G_ζ	an explanatory subgraph

$\mathbf{H}^{(l+1)}$ is the matrix of node representations at layer l, with $\mathbf{H}^{(0)} = \mathbf{X}$ being the input feature matrix. $\widetilde{\mathbf{A}}$ is the normalized adjacency matrix of an input graph G, which captures the topological feature of G. $\mathbf{W}^{(l)}$ is a learnable weight matrix at layer l (a.k.a "model weights"). σ is an activation function such as ReLU.

The *inference process* of a GNN $\mathcal{M}$ with L layers takes as input a graph $G = (X, \widetilde{\mathbf{A}})$, and computes the embedding $\mathbf{H}_v^{(L)}$ for each node $v \in V$, by recursively applying the update function in Equation 1. The final layer's output $\mathbf{H}^{(L)}$ (a.k.a "output embeddings") is used to generate a task-specific *output*, by applying a post-processing layer (*e.g.,* a softmax function). We denote the task-specific output as $\mathcal{M}(v, G)$, for the output of a GNN $\mathcal{M}$ at a node $v \in V$.

Fixed and Deterministic Inference. We say that a GNN $\mathcal{M}$ has a *fixed* inference process if its inference process is specified by fixed model parameters, number of layers, and message passing scheme. It has a *deterministic* inference process if $M(\cdot)$ generates the same result for the same input. We consider GNNs with fixed, deterministic inference processes. Such GNNs are desired for consistent and robust performance in practice.

Node Classification. Node classification is a fundamental task in graph analysis [21]. A GNN-based node classifier learns a GNN $\mathcal{M} : V \rightarrow \mathbf{Y}$ s.t. $\mathcal{M}(v) = y_v$ for $v \in V_{Tr} \subseteq V$, where V_{Tr} is the training set of nodes with known (true) labels Y_{Tr}. The inference process of a trained GNN $\mathcal{M}$ assigns the labels for a set of test nodes $V_T \subseteq V$, which are derived from their computed embeddings.

GNN Explainers and Measures. Given a GNN $\mathcal{M}$ and an output $\mathcal{M}(v, G)$ to be explained, an *explanatory subgraph* G_ζ is an edge-induced, connected subgraph of G with a non-empty edge set $E_\zeta \subseteq E$ that are responsible to clarify the occurrence of $\mathcal{M}(v, G)$. We call the set of all explanatory subgraphs as an *interpretable domain*, denoted as ζ. A GNN *explainer* is an algorithm that generates explanatory subgraphs in ζ for $\mathcal{M}(v, G)$.

An explainability measure ϕ is a function: $G_\zeta \rightarrow \mathbf{R}$ that associates an explanatory subgraph to an explainability score. Given G and an output $\mathcal{M}(v, G)$ to be explained, existing GNN explainers typically solve a single-objective optimization problem:

$$G_\zeta^* = \arg\max_{G_\zeta \in \zeta} \phi(G_\zeta) \tag{2}$$

We summarize the main notations in Table 1. We summarize in Table 2 a set of commonly adopted explainability measures, along with relevant GNN explainers.

3 SKYLINE EXPLANATIONS

We introduce our explanation structure and the generation problem.

3.1 Skyline Explanatory Query

We start with a class of explanatory queries. A *Skyline explanatory query*, denoted as SXQ, is in the form

$$\text{SXQ}(G, \mathcal{M}, V_T, \Phi)$$

where $G = (V, E)$ is an input graph, $\mathcal{M}$ is a GNN, $V_T \subseteq V$ is a set of designated test nodes of interest, and Φ is a set of user-defined explainability measures. The output $\mathcal{M}(V_T) = \bigcup_{v \in V_T}(\mathcal{M}(v, G))$ refers to the output to be explained.

Multi-objective Explanations. As aforementioned in Example 1, explanatory subgraphs that optimize a single explainability measure may not be comprehensive for the users' interpretation preference. On the other hand, a single explanatory subgraph that optimizes multiple measures may not exist, as two measures may naturally "conflict". Thus, we pursue high-quality answers for SXQ in terms of multi-objective optimality measures.

Given a node of interest $v \in V_T$, a subgraph G_ζ of G is an explanatory subgraph in the interpretable space ζ *w.r.t.* the output $\mathcal{M}(v, G)$, if it is *either* a factual *or* a counterfactual explanation. That is, G_ζ satisfies *one* of the two conditions below:

- $\mathcal{M}(v, G) = \mathcal{M}(v, G_\zeta)$;
- $\mathcal{M}(v, G) \neq \mathcal{M}(v, G \setminus G_\zeta)$

The interpretable space ζ *w.r.t. G,* $\mathcal{M}$*, and* V_T contains all the explanatory subgraphs *w.r.t.* output $\mathcal{M}(v, G)$, as v ranges over V_T.

A subset $\mathcal{G}_\zeta \subseteq \zeta$ is an *explanation w.r.t.* G, $\mathcal{M}$ and V_T if, for every node $v \in V_T$, there exists an explanatory subgraph $G_\zeta(v)$ *w.r.t.* $\mathcal{M}(v, G)$ in $\mathcal{G}$.

Explainability Measures. We make cases for three widely used explainability measures: fdl^+ measures the counterfactual property of explanatory subgraphs. Specifically, we exclude the edges of the explanatory subgraph from the original graph and conduct the GNN inference to get a new prediction based on the obtained subgraph. If the difference between these two results is significant, it indicates a good counterfactual explanatory subgraph. Similarly, fdl^- measures the factual property, i.e., how similar the explanatory subgraph is compared to the original graph in terms of getting the same predictions. conc intuitively measures how compact is the explanatory subgraph, i.e., the size of the edges.

Measurement Space. The explainability measure set Φ is a set of normalized measures to be *maximized*, each has a range $(0, 1]$. For a measure to be better minimized (*e.g.,* conciseness, fdl^- in Table 2), one can readily convert it to an inverse counterpart.

To characterize the query semantic, we introduce a dominance relation over interpretable domain ζ.

Dominance. Given a set of user-specified explanatory measures Φ (converted to *bigger* is better) and an interpretable space ζ, we say that an explanatory subgraph $G_\zeta \in \zeta$ is *dominated* by another $G'_\zeta \in \zeta$, denoted as $G_\zeta \prec G'_\zeta$, if

- for each measure $\phi \in \Phi$, $\phi(G_\zeta) \leq \phi(G'_\zeta)$; and
- there exists a measure $\phi^* \in \Phi$, such that $\phi^*(G_\zeta) < \phi^*(G'_\zeta)$.

Table 2: Representative explainability measures and notable GNN explainers

Symbol	Measure	Equation	Range	Description	Explainers
fac	factual	$\mathcal{M}(v,G) = \mathcal{M}(v,G_\zeta)$?	{true,false }	a Boolean function	[33, 35, 52, 60]
cfac	counterfactual	$\mathcal{M}(v,G) \neq \mathcal{M}(v,G \setminus G_\zeta)$?	{true,false }	a Boolean function	[33, 34, 52]
fdl$^+$	fidelity$^+$	$Pr(M(v,G)) - Pr(M(v,G \setminus G_\zeta))$	$[-1,1]$	the larger, the better	[11, 63]
fdl$^-$	fidelity$^-$	$Pr(M(v,G)) - Pr(M(v,G_\zeta))$	$[-1,1]$	the smaller, the better	[11, 35]
conc	conciseness	$\frac{1}{N}\sum_{i=1}^{N}(1 - \frac{\lvert E(G_\zeta)\rvert}{\lvert E(G)\rvert})$	$[0,1]$	the smaller, the better	[35, 63]
shapley	Shapley value	$\phi(G_\zeta) = \sum_{S \subseteq P \setminus \{G_\zeta\}} \frac{\lvert S\rvert!(\lvert P\rvert - \lvert S\rvert - 1)!}{\lvert P\rvert!} m(S, G_\zeta)$	$[-1,1]$	total contribution of nodes in G_ζ	[63]

Query Answers. We characterize the answer for an SXQ in terms of Pareto optimality. Given an interpretable space ζ *w.r.t.* G, $\mathcal{M}$, and V_T, an explanation $\mathcal{G}_\zeta \subseteq \zeta$ is a *Skyline explanation*, if

- there is no pair $\{G_1, G_2\} \subseteq \mathcal{G}_\zeta$ such that $G_1 \prec G_2$ or $G_2 \prec G_1$; and
- for any other $G \in \zeta \setminus \mathcal{G}_\zeta$, and any $G' \in \mathcal{G}_\zeta$, $G \prec G'$.

That is, $\mathcal{G}_\zeta$ is a Pareto set of ζ [50].

As a skyline explanation may still contain an excessive number of explanatory subgraphs that are too many for users to inspect, we pose a pragmatic cardinality constraint k. A k-skyline query (denoted as SXQ^k) admits, as *query answers*, skyline explanations with at most k explanatory subgraphs, or simply *k-explanations*. Here, k is a user-defined constant ($k \leq |\zeta|$]).

3.2 Evaluation of Skyline Explanatory Queries

While one can specify Φ and k to quantify the explainability of a skyline explanation, SXQ may still return multiple explanations for users to inspect. Moreover, two k-explanations, with one dominating much fewer explanations than the other in ζ, may be treated "unfairly" as equally good. To mitigate such bias, we adopt a natural measure to rank explanations in terms of dominance.

Given an explanatory subgraph $G_\zeta \in \zeta$, the dominance set of G_ζ, denoted as $\mathcal{D}(G_\zeta)$, refers to the largest set $\{G' | G' \prec G_\zeta\}$, *i.e.,* the set of all the explanatory subgraphs that are dominated by G_ζ in ζ. The *dominance power* of a k-explanation $\mathcal{G}_\zeta$ is defined as

$$\mathsf{DS}(\mathcal{G}_\zeta) = \Big| \bigcup_{G_\zeta \in \mathcal{G}_\zeta} \mathcal{D}(G_\zeta) \Big| \tag{3}$$

Note that $\mathsf{DS}(\mathcal{G}_\zeta) \leq |\zeta|$ for any explanations $\mathcal{G}_\zeta$.

Query Evaluation. Given a skyline explanatory query SXQ^k = $(G, \mathcal{M}, V_T, k, \Phi)$, the query evaluation problem, denoted as EVAL(SXQ^k), is to find a k-explanation $\mathcal{G}_\zeta^{k*}$, such that

$$\mathcal{G}_\zeta^{k*} = \underset{\mathcal{G}_\zeta \subseteq \zeta, |\mathcal{G}_\zeta| \leq k}{\arg\max} \mathsf{DS}(\mathcal{G}_\zeta) \tag{4}$$

3.3 Computational Complexity

We next investigate the hardness of evaluating skyline exploratory queries. To this end, we start with a verification problem.

Verification of Explanations. Given a query SXQ = $(G, \mathcal{M}, V_T, \Phi)$, and a set of subgraphs $\mathcal{G}$ of G, the verification problem is to decide if $\mathcal{G}$ is an explanation.

Theorem 1: *The verification problem for* SXQ *is in P.* □

Proof sketch: Given an SXQ = $(G, \mathcal{M}, V_T, k, \Phi)$ and a set of subgraphs $\mathcal{G}$ of G, we provide a procedure, denoted as Verify, that correctly determines if $\mathcal{G}$ is an explanation. The algorithm checks, for each pair (v, G_s) with $v \in V_T$ and $G_s \in \mathcal{G}$, if G_s is a factual or a counterfactual explanation of $\mathcal{M}(v, G)$. It has been verified that this process can be performed in PTIME, by invoking a polynomial time inference process of $\mathcal{M}$ for v over G_s (for testing factual explanation) and $G \setminus G_s$ (for testing counterfactual explanations), respectively [10, 45]. It outputs true if there exists a factual or counterfactual explanation for every $v \in V_T$; and false otherwise. □

While it is tractable to verify explanations, the evaluation of an SXQ is already nontrivial for $|\Phi| = 3$, even for a constrained case that $|\zeta|$ is a polynomial of $|G|$, *i.e.,* there are polynomially many connected subgraphs to be explored in G.

Theorem 2: EVAL(SXQ^k) *is* NP*-hard even when* $|\Phi| = 3$ *and* $|\zeta|$ *is polynomially bounded by* $|G|$. □

Proof sketch: The hardness of the problem can be verified by constructing a polynomial time reduction from k-representative skyline selection problem (k-RSP) [32]. Given a set S of data points, the problem is to compute a k-subset S^* of S, such that (a) S^* is a Pareto-set and (b) S^* maximizes the dominance score DS.

Given an instance of k-RSP with a set S, we construct an instance of EVAL(SXQ^k) as follows. (1) For each data point $s \in S$, create a node v_s, and construct a distinct, single-edge tree T_s at v_s. Assign a ground truth label l_s to each v_s. Let G be the union of all the single-edge trees, and define V_T as the set of root nodes of all such trees. (2) Duplicate the above set G' as a training graph G_T and train a GNN classifier $\mathcal{M}$ with layer $L \geq 1$, which gives the correct outputs. For mainstream GNNs, the training cost is in PTIME [10]. Set Φ to be a set of functions, where each $\phi \in \Phi$ assigns the i-th value of a data point s in the instance of k-RSP to be the value for the i-th explanatory measure of the matching node v_s, where $i \in [1, d]$ for d-dimensional data point in k-RSP problem. (3) Apply $\mathcal{M}$ to G with V_T as test set. Given that $\mathcal{M}$ is fixed and deterministic, the inference ensures the invariance property [22] (which generates the same results for isomorphic input G and G_T). That is, $\mathcal{M}$ assigns consistently and correctly the ground truth labels to each node v_s in G. Recall that the explanatory subgraph is connected with a non-empty edge set (§ 2). This ensures that T_s is the only factual explanation for each $v_s \in V_T$ in G. Each T_s may vary in Φ.

As $|\zeta|$ is in $O(f(|G|))$ for a polynomial function f, the above reduction is in PTIME. We can then show that there exists a k representative skyline set for k-RSP, if and only if there exists a k-explanation as an answer for the constructed instance of

EVAL(SXQ^k). As k-RSP is NP-hard for 3-dimensional space with a known input dataset, EVAL(SXQ^k) remains NP-hard for the case that $|\zeta|$ is polynomially bounded by $|G|$, and $|\Phi|$=3. □

An Exact Algorithm. A straightforward algorithm evaluates SXQ^k with exact optimal explanation. The algorithm first induces a subgraph G^L with edges within L-hop neighbors of the nodes in V_T, where L is the number of layers of the GNN $\mathcal{M}$. It then initializes the interpretable space ζ as all *connected* subgraphs in G^L. This can be performed by invoking subgraph enumeration algorithms [27, 59]. It then enumerates n Pareto sets from ζ and finds an optimal k-explanation. Although this algorithm correctly finds optimal explanations for GNNs, it is not practical for large V_T and G, as n alone can be already 2^{deg^L} (the number of connected subgraphs in G_L), and the Pareto sets to be inspected can be $\binom{n}{k}$. We thus resort to approximate query processing for SXQ, and present efficient algorithms that do not require enumeration to generate explanations that approximate optimal answers.

4 GENERATING SKYLINE EXPLANATIONS

4.1 Approximating Skyline Explanations

We introduce our first algorithm to approximately evaluate a skyline explanatory query. To characterize the quality of the answer, we introduce a notion of ϵ-explanation.

ϵ-explanations. Given explanatory measures Φ and an interpretable space ζ, we say that an explanatory subgraph $G_\zeta \in \zeta$ is *ϵ-dominated* by another $G'_\zeta \in \zeta$, denoted as $G_\zeta \preceq_\epsilon G'_\zeta$, if

- for each measure $\phi \in \Phi$, $\phi(G_\zeta) \leq (1+\epsilon)\phi(G'_\zeta)$; and
- there exists a measure $\phi^* \in \Phi$, such that $\phi^*(G_\zeta) \leq \phi^*(G'_\zeta)$.

Given a k-skyline query SXQ^k = $(G, \mathcal{M}, V_T, \Phi)$ and an interpretable domain ζ *w.r.t.* G, $\mathcal{M}$, and V_T, an explanation $\mathcal{G}_\epsilon \subseteq \zeta$ is an *(ζ, ϵ)-explanation w.r.t.* G, $\mathcal{M}$, and V_T, if (1) $|\mathcal{G}_\epsilon| \leq k$, and (2) for any explanatory subgraph $G_\zeta \in \zeta$, there is an explanatory subgraph $G'_\zeta \in \mathcal{G}_\epsilon$, such that $G_\zeta \preceq_\epsilon G'_\zeta$.

For a k-skyline query SXQ^k, a (ζ, ϵ)-explanation $\mathcal{G}_\epsilon$ properly approximates a k-explanation $\mathcal{G}_\zeta$ as its answer in the interpretable domain ζ. Indeed, (1) $\mathcal{G}_\epsilon$ has a bounded number k of explanatory subgraphs as $\mathcal{G}_\zeta$. (2) $\mathcal{G}_\epsilon$ is, by definition, an ϵ-Pareto set of ζ. In multi-objective decision making, an ϵ-Pareto set has been an established notion as a proper size-bounded approximation for a Pareto optimal set (a k-explanation $\mathcal{G}_\zeta$, in our context) [48].

(α, ϵ)-Approximations. Given a k-skyline query SXQ^k $(G, \mathcal{M}, V_T, \Phi)$, and an interpretable domain ζ *w.r.t.* G, $\mathcal{M}$, and V_T, let $\mathcal{G}^*_\zeta$ be the optimal k-explanation answer for SXQ^k in ζ (see § 3.2). We say that an algorithm is an *(α, ϵ)-approximation* for the problem EVAL(SXQ^k) *w.r.t.* ζ, if it ensures the following:

- it correctly computes an (ζ, ϵ)-explanation $\mathcal{G}_\epsilon$;
- $\mathsf{DS}(\mathcal{G}_\epsilon) \geq \alpha \mathsf{DS}(\mathcal{G}^*_\zeta)$; and
- it takes time in $O(f(|\zeta|, |G|, \frac{1}{\epsilon}))$, where f is a polynomial.

We present our main result below.

Theorem 3: *There is a $(\frac{1}{4}, \epsilon)$-approximation for* EVAL(SXQ^k) w.r.t. ζ', *where ζ' is the set of explanatory subgraphs verified by the algorithm. The algorithm computes a (ζ', ϵ)-explanation in time $O(|\zeta'|(\log \frac{r_\Phi}{\epsilon})^{|\Phi|} + L|G^L(V_T)|)$.* □

Here (1) $r_\Phi = \max \frac{\phi_u}{\phi_l}$, for each measure $\phi \in \Phi$ with a range $[\phi_l, \phi_u]$; (2) $G^L(V_T)$ refers to the set of all the L-hop neighbor subgraphs of the nodes V_T, and (3) L is the number of layers of the GNN $\mathcal{M}$. Note that in practice, $|\Phi|$, L and $\epsilon \in [0, 1]$ are small constants, and the value of r_Φ is often small.

As a constructive proof of Theorem 3, we first introduce an approximation algorithm for EVAL(SXQ^k), denoted as ApxSX-OP.

4.2 Approximation Algorithm

Our first algorithm ApxSX-OP (illustrated as Algorithm 2) takes advantage of a data locality property: For a GNN $\mathcal{M}$ with L layers and any node v in G, its inference computes $\mathcal{M}(v, G)$ with up to L-hop neighbors of v ($N^L(v)$) via message passing, regardless of how large G is. Hence, it suffices to explore and verify connected subgraphs in the L-hop neighbor subgraph $G^L(V_T)$ (see § 2). In general, it interacts with three procedures:

(1) a Generator, which initializes and dynamically expands a potential interpretable domain ζ', by generating a sequence of candidate explanatory subgraphs (or simply a "candidate") from $G^L(V_T)$;

(2) a Verifier, which asserts if an input candidate G_s is an explanatory subgraph for SXQ^k; and

(3) an Updater, that dynamically maintains a current size k (ζ', ϵ)-explanation $\mathcal{G}_\epsilon$ over verified candidates ζ', upon the arrival of verified explanatory subgraph in (2), along with other auxiliary data structures. The currently maintained explanation $\mathcal{G}_\epsilon$ is returned either upon termination (to be discussed in following parts), or upon an ad-hoc request at any time from the queryer.

Auxiliary structures. ApxSX-OP dynamically maintains the following structures to coordinate the procedures.

State Graph. ApxSX-OP coordinates the interaction of the Generator, Verifier and Updater via a state graph (simply denoted as ζ'). Each node (a "state") $s \in \zeta'$ records a candidate G_s and its local information to be updated and used for evaluating SXQ^k. There is a directed edge (a "transaction") $t = (s, s')$ in ζ', if $G_{s'}$ is obtained by applying a graph editing operator (*e.g.,* edge insertion, edge deletion) to G_s. A path ρ in the state graph ζ' consists of a sequence of transactions that results in a candidate.

In addition, each state s is associated with (1) a score $\mathsf{DS}(G_s)$, (2) a coordinate $\Phi(s)$, where each entry records an explainability measure $\phi(G_s)$ ($\phi \in \Phi$); and (3) a variable-length bitvector $B(s)$, where an entry $B(s)[i]$ is 1 if $G_i \preceq_\epsilon G_s$, and 0 otherwise. The vector $B(s)$ bookkeeps the ϵ-dominance relation between G_s and current candidates in ζ'. Its $\mathsf{DS}(G_s)$ score over ζ', can be readily counted as the number of "1" entries in $B(s)$.

"Onion Peeling". To reduce unnecessary verification, ApxSX-OP adopts a prioritized edge deletion strategy called "onion peeling". Given a node v and its L-hop neighbor subgraph $G^L(v)$, it starts with an initial state s_0 that corresponds to $G^L(V_T)$, and iteratively removes edges from the "outmost" L-th hop "inwards" to v (via Generator). This spawns a set of new candidates to be verified (by invoking Verifier), and the explanatory subgraphs are processed by Updater to maintain the explanation.

Procedure 1 Procedure updateSX

Input: a state s, a candidate G_s, state graph ζ', explanation $\mathcal{G}_\epsilon$;
Output: updated (ζ', ϵ)-explanation $\mathcal{G}_\epsilon$;
1: initializes state s with structures DS(s), $B(s)$:=$\emptyset$, $\Phi(G_s)$:=$\emptyset$;
2: evaluates $\Phi(G_s)$;
3: incrementally determines $(1+\epsilon)$-dominance of G_s;
4: updates $B(s)$ and DS(s);
5: **if** $\{G_s\}$ is a new skyline explanation **then** # New skyline
6: **if** $|\mathcal{G}_\epsilon|<k$ **then**
7: $\mathcal{G}_\epsilon := \mathcal{G}_\epsilon \cup \{G_s\}$;
8: **else**
9: $\mathcal{G}_\epsilon :=$ swap $(\mathcal{G}_\epsilon, s)$;
10: **return** $\mathcal{G}_\epsilon$.

Algorithm 2 ApxSX-OP Algorithm

Input: a query SXQ^k = $(G, \mathcal{M}, V_T, \Phi)$; a constant $\epsilon \in [0, 1]$;
Output: a (ζ', ϵ)-explanation $\mathcal{G}_\epsilon$.
1: set $\mathcal{G}_\epsilon := \emptyset$;
2: identify edges for each hop: $\mathcal{E}$=$\{E_L, E_{L-1}, \ldots, E_1\}$;
3: **for** $l = L$ to 1 **do** # Generator: Onion Peeling
4: initializes state $s_0 = G^l(V_T)$ (l-hop neighbor subgraph).
5: **while** $E_l \neq \emptyset$ **do**
6: **for** $e \in E_l$ **do**
7: spawns a state s with a candidate $G_s := G^l \backslash \{e\}$;
8: update ζ' with state s and a new transaction t;
9: **if** vrfyF(s)=False & vrfyCF(s)=False **then** # Verifier
10: continue;
11: $\mathcal{G}_\epsilon$:=updateSX $(s, G_s, \zeta', \mathcal{G}_\epsilon)$; E_l=$E_l\backslash\{e\}$; # Updater
12: **return** $\mathcal{G}_\epsilon$.

This strategy enables several advantages. (1) Observe that $\mathcal{M}(G^L(v), v) = \mathcal{M}(G, v)$ due to data locality. Intuitively, it is more likely to discover explanatory subgraphs earlier, by starting from candidates with small difference to $G^L(v)$, which is by itself a factual explanatory subgraph. (2) The strategy fully exploits the connectivity of $G^L(v)$ to ensure that the Generator produces only connected candidates with v included, over which DS, Φ, and dominance are well defined. In addition, the process enables early detection and skipping of non-dominating candidates (see "Optimization").

Algorithm. Algorithm ApxSX-OP dynamically maintains ζ' as the state graph. It first induces and verifies the L-hop neighbor subgraph $G^L(V_T)$, and initializes state node s_0 (w.r.t G_{s_0}) with $G^L(V_T)$ and local information. It then induces L-batches of edge sets E_i, $i \in [1, L]$, from $G^L(V_T)$ for onion peeling processing. For each "layer" (line 3) ($E_l, 1 \leq l \leq L$), the Generator procedure iteratively selects a next edge e to be removed from the current layer and generates a new candidate $G_{s'}$ by removing e from G_s, spawning a new state s' in ζ' with a new transaction $t = (s, s')$. Following this procedure, we obtain a "stream" of states to be verified. Each candidate is then processed by the Verifier procedures, vrfyF and vrfyCF, to test if G_s is factual or counterfactual, respectively (lines 9-10). If G_s passes the test, the associated state $s \in \zeta'$ is processed by invoking the Updater procedure updateSX, in which the coordinator $\Phi(s)$, $(1+\epsilon)$-dominance relation (encoded in $B(s)$), and DS(s) are incrementally updated (line 1). updateSX then incrementally maintains the current explanation $\mathcal{G}_\epsilon$ with the newly verified explanatory subgraph G_s, following a replacement strategy (see Procedure updateSX). The processed edge e is then removed from E_l (line 11).

Example 4: Consider the example in Figure 2. Algorithm ApxSX-OP starts the generation of explanatory subgraphs within the 2-hop subgraph s_0, by deleting one edge from the set of hop-2 edges, i.e., e_1, e_2, and e_3. This spawns three states s_1, s_2, and s_3 to be verified and evaluated. It chooses s_3 as the next state to explore, which leads to the states s_4 and e_5, in response to the deletion of e_1 and e_2, respectively. It continues to verify states s_4 and s_5. As s_4 fails the verification, i.e., it is not factual or counterfactual, it continues to verify s_5. This gives a current answer set that contains $\{s_3, s_5\}$. □

Procedure updateSX. For each new explanatory subgraph G_s (at state s), updateSX updates its information by (1) computing coordinate $\Phi(G_s)$, (2) incrementally determines if G_s is likely to join a skyline explanation in terms of $(1+\epsilon)$-dominance, *i.e.,* if for any verified explanatory subgraph $G_{s'}$ in ζ', $G_{s'} \preceq_\epsilon G_s$ (to be discussed). If so, and if the current explanation $\mathcal{G}_\epsilon$ has a size smaller than k, G_s is directly added to $\mathcal{G}_\epsilon$. (line 5- 7). Otherwise, updateSX performs a swap operator as follows: 1) identify the skyline explanation $\overline{s} \in \mathcal{G}_\epsilon$ that has the smallest $\mathcal{D}(\overline{s})$; 2) replace $\overline{s}$ with G_s, only when such replacement makes the new explanation $\mathcal{G}'_\epsilon$ having a higher score $\mathcal{D}(\mathcal{G}'_\epsilon)$ increased by a factor of $\frac{1}{k}$ (line 8- 9).

Update Dominance Relations. Procedure updateSX maintains a set of skyline explanations $\mathcal{G}_s$ (not shown) in terms of $(1+\epsilon)$-dominance. The set $\mathcal{G}_s$ is efficiently derived from the bitvectors $B(s)$ of the states in ζ'. The latter compactly encodes a lattice structure of $(1+\epsilon)$ dominance as a directed acyclic graph; and $\mathcal{G}_s$ refers to the states with no "parent" in the lattice, *i.e.,* having an explanatory subgraph that is not $(1+\epsilon)$-dominated by any others so far. We present details in full version [2].

Example 5: Recall Example 4 where the explanation space ζ' includes $\{s_1, s_2, s_3, s_5\}$. $(1+\epsilon)$-dominance is tracked by a dynamically maintained scoring table (top-right of Figure 2). As a sequence of states s_1, s_2, s_3, s_5 is generated, the first two states are verified to form a Pareto set, and added to the explanation set $\{s_1, s_2\}$. Upon the arrival of s_3, since it introduces an improvement less than a factor of $1 + \frac{1}{k} = \frac{3}{2}$, procedure updateSX skips s_3. As s_5 $(1+\epsilon)$-dominates s_1 and s_3, updateSX replaces s_1 with s_5 and updates the explanation set to be $\{s_2, s_5\}$. □

Explainability. Algorithm ApxSX-OP always terminates as it constantly removes edges from $G^L(V_T)$ and verifies a finite set of candidates. To see the quality guarantees, we present the results below.

Lemma 4: *Given a constant ϵ,* ApxSX-OP *correctly computes a (ζ', ϵ)-explanation of size k defined on the interpretation domain ζ', which contains all verified candidates.* □

Proof sketch: We show the above result with a reduction to the multi-objective shortest path problem (MOSP) [53]. Given an

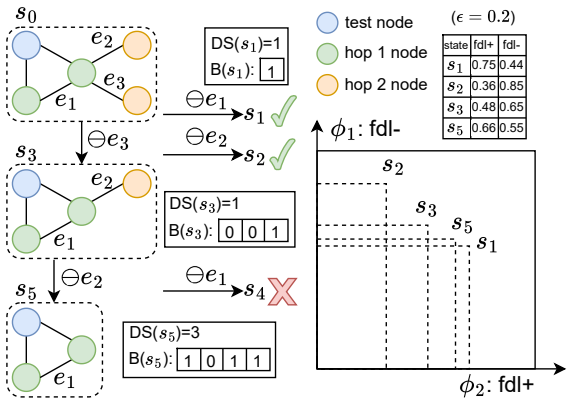


Figure 2: Illustration of Onion Peeling (L=2, k=2). $\zeta' = \{s_1, s_2, s_3, s_5\}$. A (ζ', ϵ)-explanation $\mathcal{G}_\epsilon = \{s_2, s_5\}$ with DS = 4.

edge-weighted graph G_w, where each edge carries a d-dimensional attribute vector $e_w.c$, it computes a Pareto set of paths from a start node u. The cost of a path ρ_w in G_w is defined as $\rho_w.c = \sum_{e_w \in \rho_w} e_w.c$. The dominance relation between two paths is determined by the dominance relation of their cost vector. Our reduction (1) constructs G_w as the running graph ζ' with n verified states and transitions; and (2) for each edge (s, s'), sets an edge weight as $e_w = \Phi(s) - \Phi(s')$. Given a solution Π_w of the above instance of MOSP, for each path $\rho_w \in \Pi$, we set a corresponding path ρ in ζ' that ends at a state ρ_s , and adds it into $\mathcal{G}_\epsilon$. We can verify that Π_w is an ϵ-Pareto set of paths Π_w in G_w, if and only if $\mathcal{G}_\epsilon$ is an (ζ', ϵ)-explanation of ζ'. We show that ApxSX-OP performs a more simpler process of the algorithm in [53], which ensures to generate $\mathcal{G}_\epsilon$ as a (ζ', ϵ)-explanation. □

Lemma 5: ApxSX-OP *correctly computes a (ζ', ϵ)-explanation $\mathcal{G}_\epsilon$ that ensures* $\mathsf{DS}(\mathcal{G}_\epsilon) \geq \frac{1}{4}\mathsf{DS}(\mathcal{G}_\epsilon^*)$*, where $\mathcal{G}_\epsilon^*$ is the size k (ζ', ϵ)-explanation over ζ' with maximum dominance power* $\mathsf{DS}(\mathcal{G}_\epsilon^*)$. □

Proof sketch: Consider procedure updateSX upon the arrival, at any time, of a new verified candidate G_s. (1) The above results clearly hold when $|\zeta'| \leq k$ or $|\mathcal{G}_\epsilon| \leq k$, as $\mathcal{G}_\epsilon$ is the only (ζ', ϵ)-explanation so far. (2) When $|\mathcal{G}_\epsilon| = k$, we reduce the approximate evaluation of SXQ^k to an instance of the *online MAX k-SET COVERAGE* problem [3]. The problem maintains a size-k set cover with maximized weights. We show that updateSX adopts a greedy replacement policy by replacing a candidate in $\mathcal{G}_\epsilon$ with the new candidate G_s only when this leads to a $(1 + \frac{1}{k})$ factor improvement for DS. This ensures a $\frac{1}{4}$-approximation ratio [3]. Following the replacement policy consistently, updateSX ensures a $\frac{1}{4}$ approximation ratio for (ζ', ϵ)-explanations in ζ'. □

Time Cost. As Algorithm ApxSX-OP follows an online processing for candidates generated from a stream, we provide its time cost in an output-sensitive manner, in terms of the interpretation domain ζ' when terminates. ApxSX-OP first induces the L-hop neighbor subgraph of V_T, in $O(|G^L(V_T)|)$ time. ApxSX-OP process in total $|\zeta'|$ candidates. For each candidate G_s, the Verifier process verifies G_s for SXQ (procedures vrfyF and vrfyCF), which incurs two inferences of GNN $\mathcal{M}$, in total $O(L|G^L(V_T)|)$ (assuming the number of node features are small; *cf.* Lemma 1, and [10, 45]. The total verification cost is thus in $O(|\zeta'|L|G|)$ time. For each verified explanatory subgraph G_s, the Updater procedure updateSX follows a simplified process to update a set of $(1 + \epsilon)$-Pareto set in ζ' by solving a multi-objective dominating paths problem in the state graph ζ', which takes at most $\prod_{i=1}^{|\Phi|} \left(\left\lfloor \log_{1+\epsilon} \frac{\phi_u}{\phi_l} \right\rfloor + 1\right)$ time. Here ϕ_l and ϕ_u refer to the minimum and maximum value an explainability measure $\phi \in \Phi$ has in the verified candidates in ζ'. Given ϵ is small, $\log(1 + \epsilon) \approx \epsilon$, hence the total maintenance cost is in $O\left(|\zeta'| \cdot \left(\frac{\log(r_\Phi)}{\epsilon}\right)^{|\Phi|}\right)$ time, with $r_\Phi = \frac{\phi_u}{\phi_l}$. Putting these together, the total cost is in $O(|\zeta'|(\log \frac{r_\Phi}{\epsilon})^{|\Phi|} + L|G^L(V_T)|)$ time.

Putting the above analysis together, Theorem 3 follows.

Optimization. To further reduce verification cost, ApxSX-OP uses two optimization strategies, as outlined below.

Candidate Prioritization. ApxSX-OP adopts an edge prioritization heuristic to favor promising candidates with small loss of DS and that are more likely to be a skyline explanation (Algorithm ApxSX-OP, line 7). It ranks each transaction t=(s, s') in ζ based on a loss estimation of DS by estimating $\Phi(s')$. The cost vector of each transaction is aggregated to an average weight based on each dimension, *i.e.,* $w(t) = \frac{1}{|\Phi|} \sum_{i=1}^{|\Phi|} (\phi_i(s) - \phi_i(\hat{s'}))$. The candidates from spawned states with a smallest loss of DS are preferred. This helps early convergence of high-quality explanations, and also promotes early detection of non-dominating candidates (see "early pruning").

Early Pruning. ApxSX-OP also exploits a *monotonicity property* of measures $\phi \in \Phi$ to early determine the non-ϵ-dominance of an explanatory subgraph, even before its measures are computed. Given ζ' and a measure $\phi \in \Phi$, we say ϕ is *monotonic w.r.t.* a path ρ in ζ', if for a state s with candidate G_s and another state s' with a subgraph $G_{s'}$ of G_s on the same path ρ, $\phi_l(G_s) \geq \frac{\phi_u(G_{s'})}{1+\epsilon}$, where $\phi_l(G_s)$ (resp. $\phi_u(G_{s'})$) is a lower bound estimation of $\phi(G_s)$ (resp. upper bound estimation of $\phi(G_{s'})$), *i.e.,* $\phi_l \leq \phi_l(G_s) \leq \phi(G_s)$ (resp. $\phi(G_{s'}) \leq \phi_u(G_{s'}) \leq \phi_u$). By definition, for any ϕ with monotonicity property, we have $\phi(G_{s'}) \leq \phi_u(G_{s'}) \leq (1 + \epsilon)\phi_l(G_s) \leq (1 + \epsilon)\phi(G_s)$, hence $G_{s'} \preceq_\epsilon G_s$, and any such subgraphs $G_{s'}$ of G_s can be safely pruned due to non-$(1 + \epsilon)$ dominance determined by ϕ alone, without further verification. Explainability measures such as density, total influence, conciseness, and diversity of embeddings, are likely to be monotonic. Hence, the onion peeling strategy enables early pruning by exploiting the inherent or estimated ranges of such measures. Note that the above property is checkable in $O(|\zeta'|)$ time.

Alternative Strategy: Edge Growing. As the end user may want to early termination and obtain compact, smaller-sized explanations, we also outline a variant of ApxSX-OP, denoted as ApxSX-I. It follows the same Verifier and Updater procedures, yet uses a different Generator procedure that starts with a single node v and inserts edges to grow candidate, level by level, up to its L-hop neighbor subgraph. ApxSX-I remains to be an $(\frac{1}{4}, \epsilon)$-approximation for $\mathsf{EVAL}(\mathsf{SXQ}^k)$, and does not incur additional time cost compared with ApxSX-OP. We present detailed analysis in [2].

5 DIVERSIFIED SKYLINE EXPLANATIONS

Skyline explanation may still contain explanatory subgraphs having many similar nodes. This may lead to redundant and biased explanations, as remarked earlier. Intuitively, one also prefers the explanatory subgraphs to contain various types of nodes that can clarify the model output with a more comprehensive view when inspected. We next investigate diversified SXQ evaluation.

Diversification function. Given a query $\mathsf{SXQ}^k = (G, \mathcal{M}, V_T, k, \Phi)$, the diversified evaluation problem, denoted as $\mathsf{DivEVAL}(\mathsf{SXQ}^k)$, is to find a (ζ', ϵ)-explanation $\mathcal{G}_\epsilon$, such that

$$\mathcal{G}_\epsilon^* = \underset{\mathcal{G} \subseteq \zeta', |\mathcal{G}_\epsilon| \leq k}{\arg\max} \mathsf{DivS}(\mathcal{G}_\epsilon) \tag{5}$$

where $\mathsf{DivS}(\mathcal{G}_\epsilon)$ is a *diversification function* defined on $\mathcal{G}_\epsilon$, to quantify its overall diversity. Below we introduce such a function.

As remarked earlier, we quantify the diversity of an explanation as a bi-criteria function, in terms of both neighborhood coverage and the difference between node representations, and is defined as

$$\mathsf{DivS}(\mathcal{G}_\epsilon) = \alpha \cdot \mathsf{NCS}(\mathcal{G}_\epsilon) + (1-\alpha) \cdot \sum_{G_s, G_{s'} \in \mathcal{G}_\epsilon} \mathsf{CD}(G_s, G_{s'}) \tag{6}$$

(1) NCS, a *node coverage* measure, aggregates the node coverage of explanatory subgraphs in $\mathcal{G}_\epsilon$, and (2) CD, an *accumulated difference* measure, aggregates the difference between two explanatory subgraphs. The two terms are balanced by a constant α. Specifically, we adopt a node coverage function as

$$\mathsf{NCS}(\mathcal{G}_\epsilon) = \frac{|\bigcup_{G_s \in \mathcal{G}_\epsilon} V_{G_s}|}{|V_{G^L}|}; \tag{7}$$

and for graph differences, we define CD as the accumulated Cosine distances between two explanatory graphs as:

$$\mathsf{CD}(G_s, G_{s'}) = 1 - \frac{\mathbf{x}_{G_s} \cdot \mathbf{x}_{G_{s'}}}{||\mathbf{x}_{G_s}||_2 \cdot ||\mathbf{x}_{G_{s'}}||_2} \tag{8}$$

Here, $\mathbf{x}_{G_s}$ is the embedding of G_s obtained by graph representation models such as Node2Vec [23].

Diversification Algorithm. We next outline our diversified evaluation algorithm, denoted as DivSX (pseudo-code shown in [2]). It follows ApxSX-OP and adopts onion peeling strategy to generate subgraphs in a stream. The difference is that when computing the k-skyline explanation, it includes a new replacement strategy when the marginal gain for the diversification function $\mathsf{DivS}(\cdot)$ is at least a factor of the score over the current solution $\mathsf{DivS}(\mathcal{G}_\epsilon)$. DivSX terminates when a first k-skyline explanation is found.

Procedure updateDivSX. For each new candidate G_s (state s), updateDivSX first initializes and updates G_s by (1) computing its coordinates $\Phi(G_s)$, (2) incrementally determines if G_s is a skyline explanation in terms of $(1+\epsilon)$-dominance, *i.e.,* if for any verified explanatory subgraph $G_{s'}$ in ζ', $G_{s'} \preceq_\epsilon G_s$. If so, 1) check if the current explanation $\mathcal{G}_\epsilon$ has a size smaller than k; and 2) the marginal gain of G_s is bigger than $\frac{(1+\epsilon)/2 - \mathsf{DivS}(\mathcal{G}_\epsilon)}{k - |\mathcal{G}_\epsilon|}$. If G_s satisfies both conditions, updateDivSX adds it in the k-skyline explanation $\mathcal{G}_\epsilon$.

Quality and Approximability Guarantee. Algorithm DivSX always terminates as it constantly removes edges from $G^L(V_T)$ to explore and verify a finite set of candidates. To see the quality guarantees and approximability, we show two results below.

Lemma 6: *Given a constant ϵ,* DivSX *correctly computes a (ζ', ϵ)-explanation of size k defined on the interpretation domain ζ', which contains all verified candidates.* □

The proof is similar to Lemma 4, therefore we omit it here.

Theorem 7: DivSX *correctly computes a (ζ', ϵ)-explanation $\mathcal{G}_\epsilon$ that ensures $\mathsf{DivS}(\mathcal{G}_\epsilon) \geq (\frac{1}{2} - \epsilon)\mathsf{DivS}(\mathcal{G}_\epsilon^*)$, where $\mathcal{G}_\epsilon^*$ is the size k (ζ', ϵ)-explanation over ζ' with maximum diversity power $\mathsf{DivS}(\mathcal{G}_\epsilon^*)$.* □

Proof sketch: We can verify that $\mathsf{NCS}(\cdot)$ and $\mathsf{CD}(\cdot)$ are submodular functions. Consider procedure updateDivSX upon the arrival, at any time, of a new verified candidate G_s. Given that $|\mathcal{G}_\epsilon| \leq k$ is a hard constraint, we reduce the approximate diversified evaluation of SXQ^k to an instance of the Streaming Submodular Maximization problem [4]. The problem maintains a size-k set that optimizes a submodular function over a stream of data objects. DivSX adopts a greedy increment policy by including a candidate in $\mathcal{G}_\epsilon$ with the new candidate G_s only when this leads to a marginal gain greater than $\frac{(1+\epsilon)/2 - f(S)}{k - |S|}$, where S is the current set and $f(\cdot)$ is a submodular function. This is consistent with an increment policy that ensures a $(\frac{1}{2} - \epsilon)$-approximation in [4]. □

Time Cost. Since Algorithm DivSX follows the same process as ApxSX-OP, and the update time of DivSX is $O(1)$ [4]. Therefore, according to ApxSX-OP, the time cost of DivSX is also $O(|\zeta'|(\log \frac{r_\Phi}{\epsilon})^{|\Phi|} + L|G^L(V_T)|)$.

6 EXPERIMENTAL STUDY

We conduct experiments to evaluate the effectiveness, efficiency, and scalability of our solutions. Our algorithms are implemented in Python 3.10.14 by PyTorch-Geometric framework. All experiments are conducted on a Linux system equipped with AMD Ryzen 9 5950X CPU, an NVIDIA GeForce RTX 3090, and 32 GB of RAM. **Our code and data are made available at [1]**.

6.1 Experimental Setup

Datasets. We use Cora [37], PubMed [49], FacebookPage [46], AmazonComputer [51], and OGBN_arxiv [25] (Table 3). (1) Both Cora and PubMed are citation networks with a set of papers (nodes) and their citation relations (edges). Each node has a feature vector encoding the presence of a keyword from a dictionary. For both, we consider a node classification task that assigns a paper category to each node. (2) In FacebookPage, the nodes represent verified Facebook pages, and edges are mutual "likes". The node features are extracted from the site descriptions. The task is multi-class classification, which assigns multiple site categories (politicians, governmental organizations, television shows, and companies) to a page. (3) AmazonComputeris a network of Amazon products. The nodes represent "Computer" products and an edge between two products encodes that the two products are co-purchased by the same customer. The node features are product reviews as bag-of-words. The task is to classify the product categories. (4) OGBN_arxiv is a citation network of Computer Science papers. Each paper comes with a 128-dimensional feature vector obtained by word embeddings from its title and abstract. The task is to classify the subject areas.

Table 3: Statistics of datasets

dataset	# nodes	# edges	# node features	# class labels
Cora	2,708	10,556	1,433	7
PubMed	19,717	88,648	500	3
FacebookPage	22,470	342,004	128	4
AmazonComputer	13,752	491,722	767	10
OGBN_arxiv	169,343	1,166,243	128	40

GNN Classifiers. We employ three mainstream GNNs: (1) *Graph convolutional network* (GCN) [28], one of the classic message-passing GNNs; (2) *Graph attention networks* (GAT) [54] leverage attention mechanisms to dynamically weigh the importance of a node's neighbors during inference; and (3) *Graph isomorphism networks* (GIN) [57] with enhanced expressive power up to the Weisfeiler-Lehman (WL) graph isomorphism tests.

GNN Explainers. We have implemented the following.

(1) Our skyline exploratory query evaluation methods include two approximations ApxSX-OP (§ 4.2) and ApxSX-I (§ 4.2), and the diversification algorithm DivSX (§ 5).

(2) GNNExplainer is a learning-based method that outputs masks for edges and node features by maximizing the mutual information between the probabilities predicted on the original and masked graph [60]. We induce explanatory subgraphs from the masks.

(3) PGExplainer learns edge masks to explain the GNNs. It trains a multilayer perception as the mask generator based on the learned features of the GNNs that require explanation. The loss function is defined in terms of mutual information [35].

(4) CF^2 is a GNN explainer that optimizes a linear function of weighted factual and counterfactual measures. It learns feature and edge masks, producing effective and simple explanations [52].

(5) MOExp is a model-agnostic, bi-objective optimization framework that finds Pareto optimal explanations, striking a balance between "simulatability" (factual) and "counterfactual relevance" [33].

We compare our methods with alternative explainers (§1). GNNExplainer generates a single factual exploratory subgraph, while PGExplainer emphasizes concise and factual explanation; CF^2 and MOExp optimize explanatory subgraphs based on both factuality and counterfactuality. GNNExplainer, PGExplainer, and CF^2 return only one explanatory subgraph, while MOExp does not control the number of explanatory subgraphs, therefore could return almost 200 explanatory subgraphs based on our empirical results, which are hard to inspect in practice. In contrast, users can conveniently set a bound k to output size-bounded skyline explanations by issuing an SXQ that wraps an explanation configuration.

Evaluation Metrics. For all datasets and GNNs, we select three common explainability measures (Φ): fdl^+, fdl^-, and conc. As most GNN explainers are not designed to generate explanations for multiple explainability measures, for a fair comparison, we employ three quality indicators (**QIs**) [7, 31, 38, 58]. These quality indicators are widely-used to measure how good each result is in a multi-objective manner. Consider a set of n GNN explainers, where each explainer A reports a set of explanatory subgraphs $\mathcal{G}_A$.

(1) **QI-1**: Integrated Preference Function (IPF) [8]. IPF score unifies and compares the quality of non-dominated set solutions with a weighted linear sum function. We define a normalized IPF of an explanation $\mathcal{G}_A$ from each GNN explainer A with a normalized single-objective score:

$$\text{nIPF}(\mathcal{G}_A) = \frac{1}{|\mathcal{G}_A| \cdot |\Phi|} \sum_{G \in \mathcal{G}_A} \sum_{\phi \in \Phi} \phi(G) \tag{9}$$

(2) **QI-2**: Inverted Generational Distance (IGD) [17, 31], a most commonly used distance-based QI. It measures the distance from each solution to a reference set that contains top data points with theoretically achievable "ideal" values. We introduce IGD for explanations as follows. (a) We define a universal space $\mathcal{G} = \bigcup_{i \in [1,n]} \mathcal{G}_i$ from all the participating GNN explainers, and for each explainability measure $\phi \in \Phi$, induces a reference set $\mathcal{G}^k_\phi \in \mathcal{G}$ with explanatory subgraphs having the top-k values in ϕ. (b) The normalized IGD of an explanation $\mathcal{G}_A$ from an explainer A is defined as:

$$\text{nIGD}(\mathcal{G}_A) = \frac{1}{k \cdot |\Phi|} \sum_{\phi \in \Phi} \sum_{G' \in \mathcal{G}^k_\phi} \min_{G \in \mathcal{G}_A} d(\phi(G), \phi(G')) \tag{10}$$

We use the Euclidean distance function as $d(\cdot)$ following [31, 58].

(3) **QI-3**: Maximum Spread (MS) [31]. MS is a widely-adopted spread indicator that quantifies the range of the minimum and maximum values a solution can achieve in each objective. For a fair comparison, we introduce a normalized MS score using reference sets in QI-2. For each measure $\phi \in \Phi$, and an explanation $\mathcal{G}_A$, its normalized MS score on ϕ is computed as:

$$\text{nMS}(\mathcal{G}_A)^\phi = \frac{\phi(G^{A*}_\phi)}{\phi(G^*_\phi)} \tag{11}$$

where G^*_ϕ is the explanatory subgraph with the best score on ϕ in the universal set $\mathcal{G}$, and G^{A*}_ϕ is the counterpart on ϕ from G_A.

(4) Efficiency. We report the total time cost of explanation generation. For learning-based approaches, this includes learning cost.

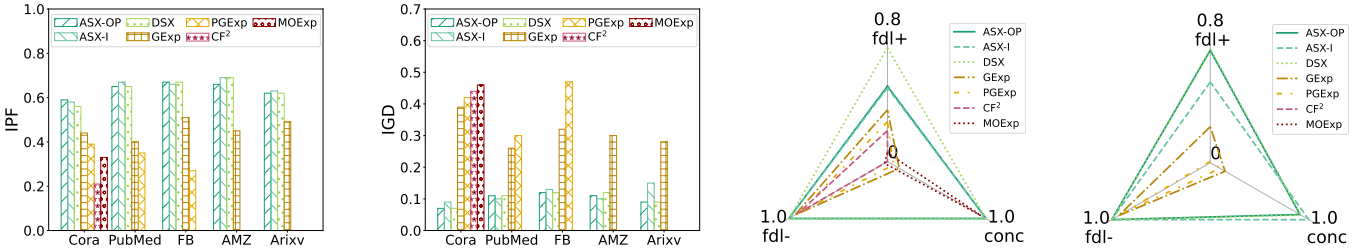
6.2 Experimental Results

We next present our findings.

Exp-1: Overall Explainability. We evaluate the overall performance of the GNN explainers using QIs.

QI-1: IPF Scores. We report IPF scores (*bigger* is better) for all GNN explainers in Figure 3(a) with GCNs and k=10. Additional explainability results with varying k are given in the full version [2]. (1) In general, ApxSX-OP outperforms a majority of competitors in aggregated explainability. For example, on Cora, ApxSX-OP outperforms GNNExplainer, PGExplainer, CF^2, and MOExp in IPF scores by 1.34, 1.51, 2.81, and 1.79 times, respectively. DivSX and ApxSX-I achieve comparable performance with ApxSX-OP. (2) We also observed that GNNExplainer and PGExplainer are sensitive as the datasets vary. For example, on FacebookPage, both show a significant increase or decrease in IPF scores. In contrast, ApxSX-OP, ApxSX-I, and DivSX consistently achieve top IPF scores over all the datasets. This verifies that our methods are quite robust in generating high-quality explanations over different data sources.

QI-2: IGD Scores. Figure 3(b) reports the IGD scores (*smaller* is better) of the explanations for GCN-based classification. ApxSX-OP, ApxSX-I, and DivSX achieve the best in IGD scores among all GNN explainers, for all datasets. This verifies that our explanation method is able to consistently select top explanations from a space of high-quality explanations that are separately optimized over different



(a) IPF Overall Score **(b) IGD Overall Score** **(c) MS on Cora** **(d) MS on PubMed**

Figure 3: Overall effectiveness of our skyline explanation compared to SOTA explainers (GNNExplainer, PGExplainer, CF^2, MOExp).

measures. In particular, we found that ApxSX-OP, ApxSX-I, and DivSX are able to "recover" the top-k global optimal solutions for each individual explainability measures with high hit rates (not shown). For example, for conc, at least 9 out of 10 are correctly and consistently identified by ApxSX-OP over every dataset.

QI-3: nMX Scores. Figures 3(c) and 3(d) visualize the normalized MS scores of the GNN explainers, over Cora and PubMed, respectively, where k = 10. (1) ApxSX-OP reports a large range and contributes to explanations that are either close or the exact optimal explanation, for each of the three measures. ApxSX-I and DivSX have comparable performance, yet with larger ranges due to its properly diversified solution. (2) DivSX, ApxSX-OP, and ApxSX-I contribute to the optimal fdl^+, fdl^-, and conc, respectively. On the other hand, each individual explainer performs worse for all measures, with a large gap. For example, in Cora, MOExp only achieves up to 3% of the best explanation (DivSX) over fdl^-.

Exp-2: Efficiency. Using the same setting as in Figure 3, we report the time cost. Figure 4(a) exhibits the following.

(1) ApxSX-OP, ApxSX-I, and DivSX outperform all (learning-based) GNN explanations. ApxSX-OP (resp. DivSX) on average outperforms GNNExplainer, PGExplainer, CF^2, and MOExp by 2.05, 4.46, 9.73, 6.81 (resp. 1.62, 3.61, 7.88, and 5.51) times, respectively. Moreover, ApxSX-OP supersedes GNNExplainer and PGExplainer better over larger graphs. CF^2 and MOExp fail to generate explanations due to high memory cost and long waiting time. Indeed, the learning costs remain their major bottleneck as for large and dense graphs.

(2) ApxSX-OP, ApxSX-I, and DivSX are feasible in generating high-quality explanations for GNN-based classification over large graphs. For example, for FacebookPage with 22,470 nodes and 342,004 edges, it takes ApxSX-OP around 150 seconds to generate skyline explanations with guaranteed quality. This verifies the effectiveness of its onion-peeling and optimization strategies.

(3) DivSX does not incur significant overhead despite that it pursues more diversified explanations. Indeed, the benefit of edge prioritization carries over to diversification process, and the incremental maintenance of the explanations reduces unnecessary verification.

(4) ApxSX-I outperforms ApxSX-OP for cases when the test nodes have "skewed" edge distribution in L-hop subgraphs, i.e., less direct neighbors but large multi-hop neighbors, which favors the edge growth strategy of ApxSX-I. ApxSX-OP takes a relatively longer time to converge to high-quality answers via onion-peeling strategy, for nodes of interest with denser neighbors at "further" hops.

Exp-3: Scalability. We report the impact of critical factors (i.e., number of explanatory subgraphs k, GNN classes, and number of GNN layers L) on the scalability of skyline explanation generation, using Cora dataset. Additional experimental results about effectiveness due to varying factors are given in the full version [2].

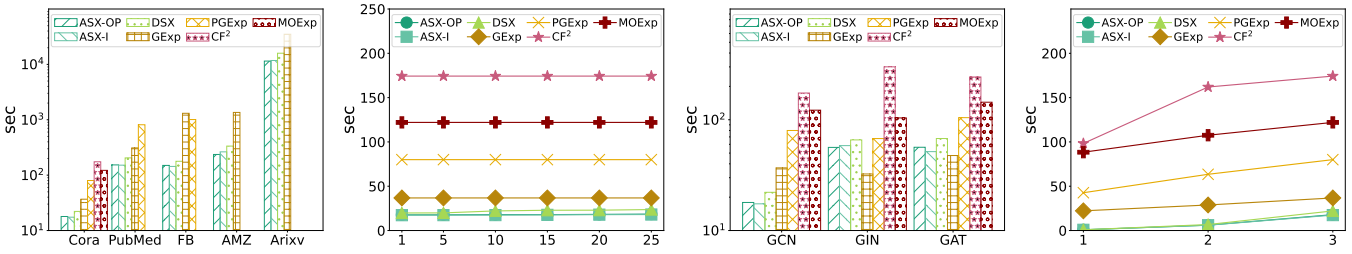
Varying k. Setting $\mathcal{M}$ as GCN-based classifier with 3 layers, we vary k from 1 to 25. Since our competitors are not configurable w.r.t. k, we show the time costs of generating their solutions (that are independent of k), along with the time cost of our ApxSX-OP, ApxSX-I, and DivSX (which are dependent on k) in Figure 4(b). Our methods take longer time to maintain skyline explanation with larger k, as more comparisons are required per newly generated candidate. DivSX is relatively more sensitive to k due to additional computation of cosine distances (§5), yet remains significantly faster than learning-based explainers. On average, our methods take up to 23 seconds to maintain the explanations with k varied to 25.

Varying GNN classes and L. Fixing k=10, we report the time cost of 3-layer GNN explainers for GCN, GAT, and GIN over Cora. As shown in Figure 4(c), all our skyline methods take the least time to explain GNN-based classification. This is consistent with our observation that the verification of GNN is the most efficient among all classes, indicating an overall small verification cost.

We fix k=10 and report the time cost of GCN explainers, with the number of layers L varied from 1 to 3 over Cora. Figure 4(d) shows us that all our methods significantly outperform the competitors. The learning overheads of competitors remain their major bottleneck, while our algorithms, as post-hoc explainers without learning overhead, are more efficient. As expected, all our methods take a longer time to generate explanations for larger L, as more subgraphs need to be verified from larger induced L-hop neighbors.

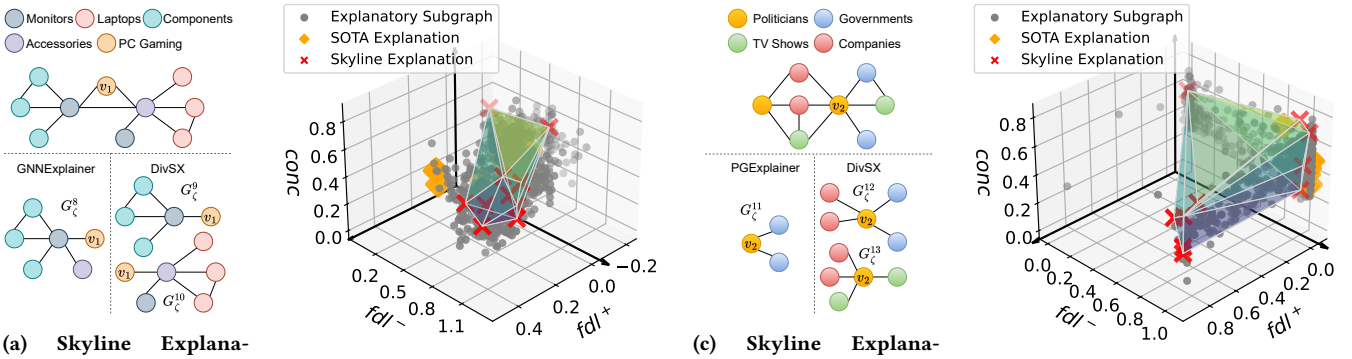
Exp-4: Case Analysis. We next showcase qualitative analyses of our explanation methods, using real-world examples from two datasets: AmazonComputer and FacebookPage.

Diversified Explanation. A user is interested in finding "Why" a product v_1 is labeled "PC Gaming" by a GCN. GNN explainers that optimize a single measure (*e.g.,* GNNExplainer) return explanations (see G_ζ^8 in Figure 5(a)) that simply reveals a fact that v_1 is co-purchased with "Components", a "one-sided" interpretation. In contrast, DivSX identifies an explanation that reveals more comprehensive interpretation with two explanatory subgraphs (both factual) $\{G_\zeta^9, G_\zeta^{10}\}$ (Figure 5(a)), which reveal two co-purchasing



(a) Efficiency (b) Scalability w. k (Cora) (c) Scalability w. GNNs (Cora) (d) Scalability w. L (Cora)

Figure 4: Efficiency and scalability of our skyline explanation vs. SOTA explainers (GNNExplainer, PGExplainer, CF^2, MOExp)



(a) **Skyline Explanation v.s.** GNNExplainer (AmazonComputer)

(b) **Visualization of Skyline Explanation** (AmazonComputer)

(c) **Skyline Explanation v.s.** PGExplainer (FacebookPage)

(d) **Visualization of Skyline Explanation** (FacebookPage)

Figure 5: Qualitative analysis of our skyline explanation compared to SOTA explainers (GNNExplainer, PGExplainer, CF^2, MOExp)

patterns bridging v_1 with not only "Components" for "Monitors" (by G_ζ^9), but also to "Accessories" of "Laptops". Indeed, a closer inspection confirms that the former indicates gamers who prefer building their gaming PC with high-refresh rate monitors designed for gaming ; and the latter indicates that v_1 is a gaming laptop which needs frequent maintenance with laptop accessories. This verifies that DivSX is able to provide more comprehensive explanation.

In Figure 5(b), we visualize the distribution of explanatory subgraphs of v_1 from Figure 5(a). Each point in the plot denotes an explanatory subgraph with 3D coordinates from normalized fdl$^+$, fdl$^-$, and conc scores. The verified interpretable space includes all the gray dots, the explanatory subgraphs generated by state-of-the-art (SOTA) explainers (i.e., GNNExplainer, PGExplainer, CF^2, MOExp) are highlighted as "diamond" points. Our skyline explanation is highlighted with red crosses. We visualize the convex hull based on skyline points to showcase their dominance over other explanatory subgraphs. We observe that the skyline explanation covers most of the interpretable space and provides comprehensive and diverse explanations. On the other hand, SOTA explainers often localize their solutions in smaller regions of the interpretable space, therefore missing diverse explanations [18].

Skyline vs. Multi-objective Explanation via Linear Combination. Our second case compares DivSX and PGExplainer (shown in Figure 5(c)). The latter generates explanations by optimizing a single objective that combines two explanation measures (factuality and conciseness). We observe that PGExplainer generates small factual subgraphs, such as G_ζ^{11} for a test node v_2, yet relatively less informative. DivSX generates a skyline explanation $\{G_\zeta^{12}, G_\zeta^{13}\}$ that is able to interpret v_2 as "Politicians" not only due to its connection with "Governments" entities but also highly related to "TV shows" and "Companies". Meanwhile, Figure 5(d) verifies that our skyline explanation covers most of the interpretable space with diverse and comprehensive explanations, while the SOTA explainers cluster their solutions in smaller regions.

We have also compared ApxSX-OP and ApxSX-I with their applications. Due to limited space, we present the details in [2].

7 CONCLUSION

We have proposed a class of skyline explanations that simultaneously optimize multiple explainability measures for GNN explanation. We have shown that the generation of skyline explanations in terms of Pareto optimality remains nontrivial even for polynomially bounded input space and three measures. We have introduced feasible algorithms for skyline explanation generation problem as well as its diversified counterpart, with provable quality guarantees in terms of Pareto optimality. Our experimental study has verified that our methods are practical for large-scale graphs and GNN-based classification and generate more comprehensive explanations, compared with state-of-the-art GNN explainers. A future topic is to develop distributed algorithms for skyline explanation generation.

REFERENCES

[1] 2024. Code and datasets. *https://github.com/DazhuoQ/SkyExp*.

[2] 2024. Full version. *https://github.com/DazhuoQ/SkyExp_full_version*.

[3] Giorgio Ausiello, Nicolas Boria, Aristotelis Giannakos, Giorgio Lucarelli, and Vangelis Th. Paschos. 2012. Online maximum k-coverage. *Discrete Applied Mathematics* 160, 13 (2012), 1901–1913.

[4] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2014. Streaming submodular maximization: massive data summarization on the fly. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 671–680.

[5] Claudio Bellei, Muhua Xu, Ross Phillips, Tom Robinson, Mark Weber, Tim Kaler, Charles E. Leiserson, Arvind, and Jie Chen. 2024. The shape of money laundering: Subgraph representation learning on the blockchain with the Elliptic2 dataset. In *KDD Workshop on Machine Learning in Finance*.

[6] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The skyline operator. In *International Conference on Data Engineering (ICDE)*. 421–430.

[7] Xinye Cai, Yushun Xiao, Miqing Li, Han Hu, Hisao Ishibuchi, and Xiaoping Li. 2021. A grid-based inverted generational distance for multi/many-objective optimization. *IEEE Transactions on Evolutionary Computation* 25, 1 (2021), 21–34.

[8] W Matthew Carlyle, John W Fowler, Esma S Gel, and Bosun Kim. 2003. Quantitative comparison of approximate solution sets for bi-criteria optimization problems. *Decision Sciences* 34, 1 (2003), 63–82.

[9] Chee-Yong Chan, H. V. Jagadish, Kian-Lee Tan, Anthony K. H. Tung, and Zhenjie Zhang. 2006. Finding k-dominant skylines in high dimensional space. In *ACM SIGMOD International Conference on Management of Data*. 503–514.

[10] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2020. Scalable graph neural networks via bidirectional propagation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[11] Tingyang Chen, Dazhuo Qiu, Yinghui Wu, Arijit Khan, Xiangyu Ke, and Yunjun Gao. 2024. View-based explanations for graph neural networks. *Proc. ACM Manag. Data* 2, 1 (2024), 40:1–40:27.

[12] Ling Cheng, Feida Zhu, Yong Wang, Ruicheng Liang, and Huiwen Liu. 2023. Evolve path tracer: Early detection of malicious addresses in cryptocurrency. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3889–3900.

[13] Kenneth Chircop and David Zammit-Mangion. 2013. On e-constraint based methods for the generation of Pareto frontiers. *Journal of Mechanics Engineering and Automation* (2013), 279–289.

[14] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: User movement in location-based social networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1082–1090.

[15] Jan Chomicki, Paolo Ciaccia, and Niccolo' Meneghetti. 2013. Skyline queries, front and back. *SIGMOD Rec.* 42, 3 (oct 2013), 6–18.

[16] Paolo Ciaccia and Davide Martinenghi. 2017. Reconciling skyline and ranking queries. *Proc. VLDB Endow.* 10, 11 (2017), 1454–1465.

[17] Carlos A Coello Coello and Margarita Reyes Sierra. 2004. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *MICAI 2004: Advances in Artificial Intelligence: Third Mexican International Conference on Artificial Intelligence. Proceedings 3*. 688–697.

[18] Indraneel Das and John E Dennis. 1997. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural optimization* 14 (1997), 63–69.

[19] Youssef Elmougy and Ling Liu. 2023. Demystifying fraudulent transactions and illicit nodes in the Bitcoin network for financial forensics. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3979–3990.

[20] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference (WWW)*. 417–426.

[21] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[22] Floris Geerts. 2023. A query language perspective on graph learning. In *ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. 373–379.

[23] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[24] Haoyu He, Yuede Ji, and H. Howie Huang. 2022. Illuminati: Towards explaining graph neural networks for cybersecurity analysis. In *IEEE European Symposium on Security and Privacy*. 74–89.

[25] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in neural information processing systems (NeurIPS)*.

[26] Ching-Lai Hwang and Abu Syed Md Masud. 1979. *Multiple objective decision making—methods and applications: a state-of-the-art survey*. Springer-Verlag.

[27] Shant Karakashian, Berthe Y Choueiry, and Stephen G Hartke. 2013. An algorithm for generating all connected subgraphs with k vertices of a graph. *Lincoln, NE* 10, 2505515.2505560 (2013).

[28] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

[29] H. T. Kung, Fabrizio Luccio, and Franco P. Preparata. 1975. On finding the maxima of a set of vectors. *J. ACM* 22, 4 (1975), 469–476.

[30] Marco Laumanns, Lothar Thiele, Eckart Zitzler, and Kalyanmoy Deb. 2002. Archiving with guaranteed convergence and diversity in multi-objective optimization. In *Genetic and Evolutionary Computation Conference (GECCO)*. 439–447.

[31] Miqing Li and Xin Yao. 2019. Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys (CSUR)* 52, 2 (2019), 1–38.

[32] Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. 2006. Selecting stars: The k most representative skyline operator. In *International Conference on Data Engineering (ICDE)*. 86–95.

[33] Yifei Liu, Chao Chen, Yazheng Liu, Xi Zhang, and Sihong Xie. 2021. Multi-objective explanations of GNN predictions. In *IEEE International Conference on Data Mining (ICDM)*. 409–418.

[34] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. 2022. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 4499–4511.

[35] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized explainer for graph neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[36] R Timothy Marler and Jasbir S Arora. 2010. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization* 41 (2010), 853–862.

[37] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3 (2000), 127–163.

[38] Alejandro Morales-Hernández, Inneke Van Nieuwenhuyse, and Sebastian Rojas Gonzalez. 2022. A survey on multi-objective hyperparameter optimization algorithms for machine learning. *Artif. Intell. Rev.* 56, 8 (2022), 8043–8093.

[39] Shiyu Ouyang, Qianlan Bai, Hui Feng, and Bo Hu. 2024. Bitcoin money laundering detection via subgraph contrastive learning. *Entropy* 26, 3 (2024).

[40] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2003. An optimal and progressive algorithm for skyline queries. In *ACM SIGMOD International Conference on Management of Data*. 467–478.

[41] Christos H. Papadimitriou and Mihalis Yannakakis. 2000. On the approximability of trade-offs and optimal access of Web sources. In *Annual Symposium on Foundations of Computer Science, (FOCS)*. 86–92.

[42] Peng Peng and Raymond Chi-Wing Wong. 2018. Skyline queries and pareto optimality. In *Encyclopedia of Database Systems, Second Edition*. Springer.

[43] Nadia Pocher, Mirko Zichichi, Fabio Merizzi, Muhammad Zohaib Shafiq, and Stefano Ferretti. 2023. Detecting anomalous cryptocurrency transactions: An AML/CFT application of machine learning-based forensics. *Electronic Markets* 33, 1 (2023), 37.

[44] Sudeep Pushpakom, Francesco Iorio, Patrick A Eyers, K Jane Escott, Shirley Hopper, Andrew Wells, Andrew Doig, Tim Guilliams, Joanna Latimer, Christine McNamee, Alan Norris, Philippe Sanseau, David Cavalla, and Munir Pirmohamed. 2019. Drug repurposing: progress, challenges and recommendations. *Nature reviews Drug discovery* 18, 1 (2019), 41–58.

[45] Dazhuo Qiu, Mengying Wang, Arijit Khan, and Yinghui Wu. 2024. Generating robust counterfactual witnesses for graph neural networks. In *IEEE International Conference on Data Engineering (ICDE)*. 3351–3363.

[46] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021).

[47] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. 2020. Interpreting graph neural networks for NLP with differentiable edge masking. In *International Conference on Learning Representations (ICLR)*.

[48] Oliver Schütze and Carlos Hernández. 2021. Computing ϵ-(approximate) pareto fronts. In *Archiving Strategies for Evolutionary Multi-objective Optimization Algorithms*. 41–66.

[49] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* 29, 3 (2008), 93–106.

[50] Shubhkirti Sharma and Vijay Kumar. 2022. A comprehensive review on multi-objective optimization techniques: Past, present and future. *Archives of Computational Methods in Engineering* 29, 7 (2022), 5605–5633.

[51] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. In *Relational Representation Learning Workshop @ NeurIPS*.

[52] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. 2022. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *ACM web conference (WWW)*. 1018–1027.

[53] George Tsaggouris and Christos Zaroliagis. 2009. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications.

Theory of Computing Systems 45, 1 (2009), 162–186.
[54] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations (ICLR)*.
[55] Yuyang Wang, Zijie Li, and Amir Barati Farimani. 2023. *Graph neural networks for molecules.* Springer International Publishing, 21–66.
[56] Lanning Wei, Huan Zhao, Zhiqiang He, and Quanming Yao. 2023. Neural architecture search for GNN-based graph classification. *ACM Trans. Inf. Syst.* 42, 1 (2023).
[57] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *International Conference on Learning Representations (ICLR)*.
[58] Ke Xue, Jiacheng Xu, Lei Yuan, Miqing Li, Chao Qian, Zongzhang Zhang, and Yang Yu. 2022. Multi-agent dynamic algorithm configuration. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[59] Zhengyi Yang, Longbin Lai, Xuemin Lin, Kongzhang Hao, and Wenjie Zhang. 2021. HUGE: An efficient and scalable subgraph enumeration system. In *International Conference on Management of Data (SIGMOD)*.
[60] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[61] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. 2018. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[62] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2023. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 5 (2023), 5782–5799.
[63] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning (ICLR)*.
[64] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in neural information processing systems (NerIPS)*.
[65] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. 2022. Gstarx: Explaining graph neural networks with structure-aware cooperative games. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[66] Youmin Zhang, Qun Liu, Guoyin Wang, William K Cheung, and Li Liu. 2024. GEAR: Learning graph neural network explainer via adjusting gradients. *Knowledge-Based Systems* 302 (2024), 112368.

APPENDIX

7.1 Additional Algorithmic Details

Update Dominance Relationships. We introduce the details of Procedure updateSX. With the stream of upcoming states, i.e., explanatory subgraphs, we maintain a lattice structure for the dominance relationships. Specifically, we first compare the new state s with the current skylines: 1) If certain skylines dominate it, we connect a directed edge from these dominating skylines to the new state, respectively. 2) If the new state dominates certain skylines, we connect directed edges from the new state to these dominated skylines. Meanwhile, we reconnect the dominant states of these skylines to the new state. 3) If the new state is not dominated by any skylines, then we obtain its $\mathcal{D}(s)$ using the bitvector $B(s)$. Next, we identify the skyline $\bar{s}$ with the smallest $\mathcal{D}(\bar{s})$, replace $\bar{s}$ with s, only when such replacement makes the $\mathcal{D}$ increased by a factor of $\frac{1}{k}$. If swapping is conducted, the new skyline, i.e. s, connects directed edges to the dominating states based on bitvector $B(s)$, and we remove the swapped skyline and its corresponding edges. The above swap strategy ensures a $\frac{1}{4}$-approximation ratio [3].

Alternative Strategy: Edge Growing. Edge growing strategy differs from the "onion peeling" strategy with the test node as the initial state and the L-hop neighbor subgraph as the final state. ApxSX-I adopts Breadth-First Search to explore explanatory subgraphs. Specifically, starting from the test node, we first connect the test node with each of its neighbors, respectively. Then we verify these candidate subgraphs and determine the $(1+\epsilon)$-dominance relationships. Following the same candidate prioritization as ApxSX-OP, we select the neighboring node with the biggest weight. Then, continuing with this newly chosen node, we explore its neighbors, except the ones that we previously visited. We expand the subgraph with the neighbors, respectively, and conduct candidate prioritization again. We iteratively continue this process until we explore the explanatory subgraph as the L- neighbor subgraph. The search space and time complexity of ApxSX-I remains the same as ApxSX-OP, since the interpretation domain ζ' and the update strategy are the same.

Algorithm 3 DivSX Algorithm

Input: a query $\text{SXQ}^k = (G, \mathcal{M}, V_T, \Phi)$; a constant $\epsilon \in [0, 1]$;
Output: a (ζ', ϵ)-explanation $\mathcal{G}_\epsilon$.

```
set G_ε := ∅;
identify edges for each hop: E={E_L, E_{L−1}, ..., E_1};
for l = L to 1 do                          # Generator: Onion Peeling
    initializes state s_0 = G^l(V_T) (l-hop neighbor subgraph).
    while E_l ≠ ∅ do
        for e ∈ E_l do
            spawns a state s with a candidate G_s := G^l \ {e};
            update ζ' with state s and a new transaction t;
            if vrfyF(s)=False & vrfyCF(s)=False then    # Verifier
                continue;
            G_ε :=updateDivSX (s, G_s, ζ', G_ε); E_l=E_l\{e}; # Updater
return G_ε.
```

Procedure 4 Procedure updateDivSX

Input: a state s, an explanatory subgraph G_s, state graph ζ'; an explanation $\mathcal{G}_\epsilon$;
Output: Updated (ζ', ϵ)-explanation $\mathcal{G}_\epsilon$;

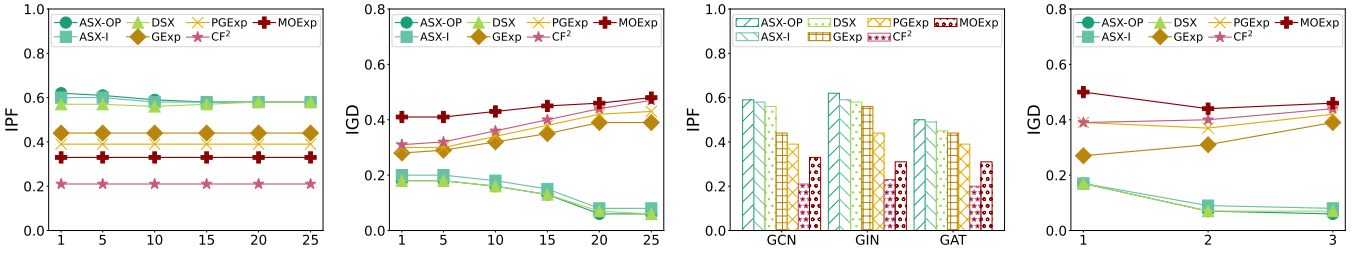
```
initializes state s with structure Φ(G_s):=∅;
evaluates Φ(G_s);
incrementally determines (1 + ε)-dominance of G_s;
if {G_s} is a new skyline explanation then      # New skyline
    Δ(s|G_ε) := DivS(G_ε ∪ {G_s}) − DivS(G_ε)
    if |G_ε|<k and Δ(s|G_ε) ≥ ((1+ε)/2 − DivS(G_ε)) / (k − |G_ε|) then
        G_ε := G_ε ∪ {G_s};
return G_ε.
```

Diversification Algorithm. The detailed pseudo-codes of diversification algorithm DivSX are shown as Algorithm 3 and Procedure 4. Please find the detailed descriptions of DivSX in § 5.

7.2 Additional Experimental Results

Impact of Factors. We report the impact of critical factors, i.e., size of explanations k, GNN classes, and the number of layers L, on the effectiveness of generated skyline explanation over the Cora dataset.

Impact of k. Setting $\mathcal{M}$ as GCN-based classifier with 3 layers, we vary k from 1 to 25. Since our competitors are not configurable w.r.t. k, we show the IPF and IGD scores of their solutions (that are independent of k), along with our ApxSX-OP, ApxSX-I, and DivSX



(a) IPF varying k (b) IGD varying k (c) IPF varying GNNs (d) IGD varying L

Figure 6: Impact of factors on the effectiveness of explanations

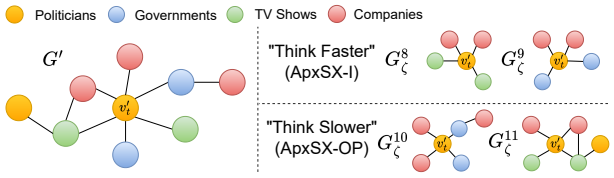


Figure 7: Different edge explore strategies show different preferences for returned skyline explanation.

(which are dependent on k) in Figures 6(a) and 6(b). As k becomes larger, ApxSX-OP, ApxSX-I, and DivSX all achieve better scores and are able to consistently generate explanations with higher quality. This is because larger k allows more dominating subgraphs to be verified, whenever possible for at least one measure. In contrast, our competitors are not aware of the growth of k, given that they may "stuck" at local optimal explanations that are optimized over a single measure, hence fail to improve the solution even with larger k. Moreover, MOExp does not explicitly constrain the size of the explanation set, which can result up to 200 explanatory subgraphs based on our empirical results. Meanwhile, even with k=1, our three methods achieve the highest scores in both IPF and IGD, compared to our competitors.

Varying GNN *classes.* Fixing k=10, we report IPF scores of 3-layer GNN explainers for representative GNN classes including GCN, GAT, and GIN. As verified in Figure 6(c), ApxSX-OP, ApxSX-I, and DivSX consistently outperform other explanations for different GNN classes. This verifies that our approaches remain stable in generating high-quality explanations for different types of GNNs. Our observations for IGD scores are consistent, hence omitted.

Varying number of layers. Fixing k=10, we report IGD scores of GCN explainers, with the number of layer L varies from 1 to 3. over Cora. Fig 6(d) shows that ApxSX-OP, ApxSX-I, and DivSX are able to achieve better IGD scores for generating explanations with more complex GNNs having "deeper" architecture. Indeed, as L becomes larger, our all three methods are able to identify more connected subgraphs that contribute to better fdl^+ and fdl^- scores, hence improving explanation quality.

Case Study: Insertion v.s. Onion-Peeling. In this case study, we compare the application scenarios of ApxSX-OP and ApxSX-I, as shown in Figure 7. We observe that ApxSX-OP, with onion peeling, cater users' needs who "think slow" and seeks in-depth explanations that are counterfactual. Such larger and counterfactual explanations are more likely to affect GNN behaviors if removed [33]. ApxSX-I, on the other hand, due to its containment of edges closer to targeted ones via edge insertions, tends to produce – at its early stage – smaller, factual explanations as convincing evidences. This caters users who "think fast" and is good at quick generation of factual explanations [33]. For example, in answering "why" node v_t' is recognized as a "Politician" site, G_ζ^8 and G_ζ^9 are among skyline explanations from ApxSX-I at early stage, revealing the fact that v_t' frequently interact with TV shows, Governments, and Companies, covering its ego-networks as quick facts. G_ζ^{10} and G_ζ^{11}, which are reported by ApxSX-OP in its early stage via "onion-peeling", are relatively larger counterfactual explanations that suggest interaction patterns with more complex structures. For example, G_ζ^{11} includes two-hop neighbors and a triangle, which indicates the importance of the corresponding company and TV show. Most importantly, another politician's two-hop neighbor also shares these nodes, revealing the close relationship between the two politicians.