# Interpreting Graph Inference with Skyline Explanations


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

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

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

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



***Abstract*—Inference queries have been routinely issued to graph machine learning models such as graph neural networks (GNNs) for various network analytical tasks. Nevertheless, GNN outputs are often hard to interpret comprehensively. Existing methods typically conform to individual pre-defined explainability measures (such as fidelity), which often leads to biased, "one-sided" interpretations. This paper introduces *skyline explanation*, a new paradigm that interprets GNN outputs by simultaneously optimizing multiple explainability measures of users' interests. (1) We propose skyline explanations as a Pareto set of explanatory subgraphs that dominate others over multiple explanatory measures. We formulate skyline explanation as a multi-criteria optimization problem, and establish its hardness results. (2) We design efficient algorithms with an onion-peeling approach, which strategically prioritizes nodes and removes unpromising edges to incrementally assemble skyline explanations. (3) We also develop an algorithm to diversify the skyline explanations to enrich the comprehensive interpretation. (4) We introduce efficient parallel algorithms with load-balancing strategies to scale skyline explanation for large-scale GNN-based inference. Using real-world and synthetic graphs, we experimentally verify our algorithms' effectiveness and scalability.**


## I. Introduction

Graph models such as graph neural networks (GNNs) have been trained and routinely queried to perform network analysis in, *e.g.,* biochemistry, social and financial networks [1]–[4], among other graph analytical tasks. Given a graph $G$ and a set of test nodes $V_T$ in $G$, a GNN $\mathcal{M}$ can be considered as a function that converts a feature representation of $G$ in the form of $(X, A)$, where $X$ (resp. $A$) refers to the node feature matrix (resp. normalized adjacency matrix) of $G$, to an output representation $Z$ ("output embedding matrix"). The output $Z$ can be post-processed to task-specific output such as labels for classification. An "inference query" invokes the above process to query $\mathcal{M}$, to perform the inference analysis of $G$.

Despite the promising performance of GNNs, it remains desirable yet nontrivial to interpret their outputs to help users understand GNN-based decision making [5]. Several methods ("explainers") have been proposed [6]–[10]. Typically, a GNN explainer extracts a subgraph $G_\zeta$ of $G$ that can best clarify the output of an inference query posed on $\mathcal{M}$ over $G$. This is often addressed by discovering a subgraph $G_\zeta$ subject to a pre-defined metric, which quantifies the explainability of $G_\zeta$ for the output (as summarized in Table I).

For example, a subgraph $G_\zeta$ 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 $\mathcal{M}$) [6], [8], [10], [11]. $G_\zeta$ is a "*counterfactual*" explanation, if removing the edges of $G_\zeta$ from $G$ leads to a change of the output of $\mathcal{M}$ on the remaining graph (denoted as $G \setminus G_\zeta$) [7], [8], [11]. Other metrics include fidelity$^-$ [7], [8], [11] (resp. fidelity$^+$ [10], [12]), which quantifies the explainability of $G_\zeta$ in terms of the closeness between the task-specific output of $\mathcal{M}$, such as the probability of label assignments, over $G_\zeta$ (resp. $G \setminus G_\zeta$) and their original counterpart $G$, and "*conciseness* (*sparsity*)" that favors small explanatory subgraphs.

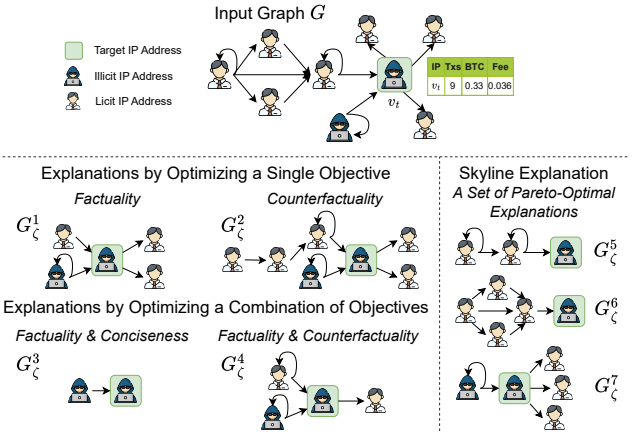



Fig. 1: A Bitcoin transaction network with a target IP address (test node $v_t$) that has a label "*illicit*" to be explained [13].

Nevertheless, such a metric assesses explanations from "one-sided" perspective of explainability (§ II). 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:** Figure 1 illustrates a fraction of a Bitcoin blockchain transaction network $G$. Each IP address (node) is associated with transactions (edges) and transaction features, such as frequency of transactions, the amount of transacted Bitcoins, and the amount of fees in Bitcoins. A GNN-based classifier $\mathcal{M}$ is trained to detect illicit IP addresses, by assigning a label (*e.g.,* "illicit") to nodes (*e.g.,* $v_t$).

A law enforcement agency has posed an inference query over $\mathcal{M}$ to get the output, and wants to further understand "why" the GNN asserts $v_t$ as an illicit account address.

TABLE I: 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 | [6], [8], [10], [11] |
| cfac | counterfactual | $\mathcal{M}(v,G) \neq \mathcal{M}(v,G \setminus G_\zeta)$? | {true,false } | a Boolean function | [7], [8], [11] |
| fdl$^+$ | fidelity$^+$ | $Pr(M(v,G)) - Pr(M(v,G \setminus G_\zeta))$ | $[-1,1]$ | the larger, the better | [12], [14] |
| fdl$^-$ | fidelity$^-$ | $Pr(M(v,G)) - Pr(M(v,G_\zeta))$ | $[-1,1]$ | the smaller, the better | [10], [14] |
| 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 | [10], [12] |
| 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 $G_\zeta$ | [12] |

They may further ask clarifications by asking "*which fraction of the graph $G$ is responsible for the* GNN*'s decision on assigning the label "illicit" to the account $v_t$?*", and ground this output by referring to known real-world money laundering scenarios [15], [16]). For example, "*Spindle*" [15] suggests that perpetrators generate multiple shadow addresses to transfer small amounts of assets along lengthy paths to a specific destination; and *Peel Chain* [16] 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.

An explanatory subgraph $G_\zeta^1$ is a factual explanatory subgraph generated by the explainer in [6]; and $G_\zeta^2$ is a counterfactual one generated by [7]. Comparing $G_\zeta^1$ with $G_\zeta^2$, we observe the following. (1) $G_\zeta^1$ is a small and concise explanation that only involves $v_t$ and most of its neighbors, which includes neighborhood features that preserve the output of a GNN's output; nevertheless, it misses important nodes that have a higher influence on GNN's output, that are not necessarily in $v_t$'s direct neighborhood. (2) Such nodes can be captured by a counterfactual explanatory subgraph, as depicted in $G_\zeta^2$. However, a larger fraction of $G$ is included to ensure that the "removal" of edges incurs enough impact to change the output of the GNN, hence sacrificing "conciseness", conflicting with users' need on quick generation of small evidence which is "faithful" to the original GNN output [11].

Moreover, choosing either alone can be biased to a "one-sided" explanation for the "illicit" IP address, grounded by one of the two money laundering patterns, but not both. □

Can we generate explanations that *simultaneously* address multiple explainability metrics? A quick solution is to compute subgraphs that optimize a weighted linear combination of all metrics [17], [18]. However, this may lead to a marginally optimal answer over all measurements, overlooking other high-quality and diverse solutions, hence an overkill.

**Example 2:** Consider another two explanatory subgraphs: $G_\zeta^3$ is the explanation generated by [10], an explainer that optimizes both conciseness and factual measures [10]; $G_\zeta^4$ is from an explainer that linearly combines factual and counterfactual measures into a single, bi-criteria objective function to be optimized [8]. Such methods enforce 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 fidelity$^-$ and conciseness both encourage smaller and faithful explanations, explanatory subgraphs obtained by [10] such as $G_\zeta^3$ turns out to be relatively much smaller and less informative, hardly providing sufficient real-world evidence that can be used to interpret money laundering behaviors. On the other hand, explanations from [8] such as $G_\zeta^4$ may easily be a one-sided, factual or counterfactual evidence. We found that such explanations capture only one type of money laundering scenario at best in most cases. □

Skyline queries [19]–[23] compute Pareto sets ("skylines") that dominate the rest data points across a set of quality measures. Skylines generally offer better and more comprehensive solutions against the aforementioned alternatives [18], [24]. We advocate approaching GNN explanation by generating a set of subgraphs that are Poreto sets over multiple user-defined explanatory measures, referred to as "*skyline explanations*".

**Example 3:** Consider a "skyline query" that seeks subgraphs addressing multiple explainability criteria, which returns a set of subgraphs $G_\zeta^5$, $G_\zeta^6$, and $G_\zeta^7$. 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, $G_\zeta^5$, $G_\zeta^6$, and $G_\zeta^7$ capture different money laundering scenarios: *Peel Chain* [16], *Spindle* [15], and a combination, respectively. Therefore, such a skyline explanation supports law agencies by revealing a more comprehensive interpretation of the decision-making of GNNs on $v_t$. □

We advocate for developing 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 interest 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 [25], cybersecurity analysis [26], fraud detection [27], social recommendation [28], among others, where GNN output should be clarified from multiple, comprehensive aspects, rather than one-sided, biased perspectives.

**Contribution**. We summarize our contribution as follows.
(1) *A formulation of Skyline Explanation*. 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 $\mathcal{M}$, a node of interest $v_t$, and a set of

explainability measures $\Phi$, and requests a skyline explanation (a set of explanatory subgraphs of size $k$) that clarifies the output of $\mathcal{M}$ over $v_t$, which simultaneously optimizes the measures in $\Phi$. We introduce a subgraph dominance relation over $\Phi$, and verify the hardness of the problem.

(2) We introduce efficient algorithms to evaluate SXQs. (i) Our first algorithm adopts an "onion-peeling" strategy to iteratively reduce edges at each hop of targeted nodes, and validates a bounded number of generated explanations in a discretized coordination system, to incrementally improve the explainability of the subgraphs. We show that this process ensures a $(1+\epsilon)$ approximation of the optimal solution. (ii) We also present an algorithm to diversify the answer set for SXQs, to provide a comprehensive solution in terms of node embedding and structural differences. (iii) Additionally, we propose parallel algorithms that scale to large graphs by clustering workloads to optimize edge information sharing and load balancing.

(3) Using real-world graphs and benchmark tasks from various domains, we experimentally verify that our approach outperforms state-of-the-art explainers, generates more comprehensive interpretation, and scales well to billon-scale graphs. For example, it outperforms CF$^2$ [8] in the Integrated Preference score by 2.8 times on Cora dataset; for OGBN_arxiv dataset with over one million edges, it improves the fastest baseline GExp [6] by 8.2 times.

**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. Recent studies proposed multiple variants of the GNNs, such as graph convolution networks (GCNs) [29], graph attention networks (GATs) [30], Graph Isomorphism Networks (GINs) [31]. These methods generally follow an information aggregation scheme where features of a node are obtained by aggregating features from 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 [6] 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 [7] learns the counterfactual subgraphs, which lead to significant changes of the output if removed from the graphs. PGExplainer [10] parameterizes the learning process of mask matrix using a multi-layer perceptron. GraphMask [32] 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 require prior knowledge of model parameters and incur learning overhead for large graphs. Post-hoc approaches perform post-processing to directly compute subgraphs that optimize a fixed criteria. For example, [12] utilizes Monte-Carlo tree search to compute subgraphs that optimize a game-theory-inspired Shapley value. [33] follows a similar approach, yet optimizes HN values, a topology-aware variant of Shapley value. These methods are constrained to pre-defined criterion, and cannot easily adapt to customizable, multiple explanability criteria.

Closer to our work are GNN explainers that optimize pre-defined, multiple criteria. CF$^2$ [8] optimizes a linear function of factual and counterfactual measures and learns feature and edge masks as explanations. RoboGExp [34] generates factual, counterfactual, and robust subgraphs, which remain unchanged upon a bounded number of edge modifications. GEAR [9] learns GNN explainers by adjusting the gradients of multiple objectives geometrically during optimization. MOExp [11] solves a bi-objective optimization problem to find Pareto optimal set over "simulatability" (factual) and "counterfactual relevance". Despite these methods generating explanations over multiple criteria, the overall goal remains pre-defined – and not configurable as needed. In addition, MOExp does not control the size of explanations, which may result in large number of explanations that are hard to inspect.

## II. Graphs and GNN Explanation

**Graphs.** A 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 $G$, the *$L$-hop neighbors* of $v$, denoted as $N^L(v)$, refers to the set of all the nodes in the $L$-hop of $v$ in $G$. The *$L$-hop subgraph* of a set of nodes $V_s \subseteq V$, denoted as $G^L(V_s)$, is the subgraph induced by the node set $\bigcup_{v_s \in V_s} N^L(v_s)$.

**Graph Neural Networks.** GNNs [29], [35], [36] 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.

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

$\mathbf{H}^{(l+1)}$ is the matrix of node representations at layer $l$, with $\mathbf{H}^{(0)} = \mathbf{X}$ being the input feature matrix. $\widetilde{\mathbf{A}}$ is a 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"). $\sigma$ is an activation function.

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 for consistent and robust performance in practice.

TABLE II: 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 $v \in V$ |
| $v_t$; $V_T$ | a (test) node; a set of (test) nodes |
| $s$; $G_s$; $G_\zeta$ | a state $s$; a candidate $G_s$; an explanatory subgraph |

**Node Classification.** Node classification is a fundamental task in graph analysis [37]. A GNN-based node classifier learns a GNN $\mathcal{M} : (X, \mathbf{A}) \rightarrow \mathbf{Y}$ 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 label for a test node $v_t \subseteq V_T$, 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* $G_\zeta$ 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 space*, denoted as $\zeta$.

A GNN *explainer* is an algorithm that generates explanatory subgraphs in $\zeta$ for $\mathcal{M}(v, G)$. An *explainability measure* $\phi$ is a function: $G_\zeta \rightarrow \mathbf{R}$ that associates an explanatory subgraph with an explainability score. Given $G$ and $\mathcal{M}(v, G)$ to be explained, existing explainers typically solve a single-objective optimization problem: $G_\zeta^* = \arg\max_{G_\zeta \in \zeta} \phi(G_\zeta)$.

Several GNN explainers and the explanability measures they aim to optimize are summarize in Table I. We summarize the main notations in Table II.

## III. Skyline Explanations

As aforementioned in Example 1, explanations that optimize a single explainability measure may not be comprehensive for users' interpretation preference. Meanwhile, a single explanation that optimizes multiple measures may not exist, as two measures may naturally "conflict". Thus, we pursue high-quality explanations for SXQ in terms of multi-objective optimality measures. We next introduce our skyline explanation structure and the corresponding generation problem.

### A. Skyline Explanatory Query

We start with a class of explanatory queries. A *Skyline explanatory query*, denoted as SXQ, has a form $\mathsf{SXQ}(G, \mathcal{M}, v_t, \Phi)$, where (1) $G = (V, E)$ is an input graph, (2) $\mathcal{M}$ is a GNN; (3) $v_t \subseteq V_T$ is a designated test node of interest, with an output $\mathcal{M}(v_t, G)$ to be explained; and (4) $\Phi$, the *measurement space*, refers to a set of normalized explanability measures to be *maximized*, each has a range $(0, 1]$. For a measure to be better minimized (*e.g.,* conc, $\mathsf{fdl}^-$ in Table I), one can readily convert it to an inverse counterpart.

To characterize the query semantic, we specify explanatory subgraphs and their dominance relation.

*Explanatory Subgraphs*. Given a node of interest $v_t \in V_T$, a subgraph $G_\zeta$ of $G$ is an *explanatory subgraph w.r.t.* the output $\mathcal{M}(v_t, G)$, if it is *either* a factual *or* a counterfactual explanation. That is, $G_\zeta$ satisfies *one* of the two conditions:
- $\mathcal{M}(v_t, G) = \mathcal{M}(v_t, G_\zeta)$;
- $\mathcal{M}(v_t, G) \neq \mathcal{M}(v_t, G \setminus G_\zeta)$

The interpretable space $\zeta$ *w.r.t.* a skyline explanatory query $\mathsf{SXQ}(G, \mathcal{M}, v_t, \Phi)$ contains all the explanatory subgraphs *w.r.t.* output $\mathcal{M}(v_t, G)$.

*Dominance*. Given a measurement space $\Phi$ and an interpretable space $\zeta$, 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 $\phi^*(G_\zeta) < \phi^*(G'_\zeta)$.

*Query Answers*. Given $\mathsf{SXQ}(G, \mathcal{M}, v_t, \Phi)$ with an interpretable space $\zeta$, a set of explanatory subgraphs $\mathcal{G}_\zeta \subseteq \zeta$ 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 [24] of the interpretable space $\zeta$.

A skyline explanation may contain an excessive number of subgraphs that are too many to inspect. We thus pose a pragmatic cardinality constraint $k$. We say a skyline explanation is a *$k$-explanation*, if it contains at most $k$ explanatory subgraphs ($k \leq |\zeta|$]). A $k$-skyline query (denoted as $\mathsf{SXQ}^k$) admits only $k$-explanations as its query answers.

### B. Quality of Skyline Explanations

A GNN output may still have multiple $k$-explanations that can be too many for users to inspect. Moreover, one $k$-explanation dominating significantly less explanations than another in $\zeta$, may be treated "unfairly" as equally good. To mitigate such bias, we introduce a notion of dominance power.

Given an explanatory subgraph $G_\zeta \in \zeta$, the dominance set of $G_\zeta$, 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 $G_\zeta$ in $\zeta$. The *dominance power* of a $k$-explanation $\mathcal{G}_\zeta$ is defined as

$$\mathsf{DS}(\mathcal{G}_\zeta) = \left| \bigcup_{G_\zeta \in \mathcal{G}_\zeta} \mathcal{D}(G_\zeta) \right| \tag{2}$$

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

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

$$\mathcal{G}_\zeta^{k*} = \underset{\mathcal{G}_\zeta \subseteq \zeta, |\mathcal{G}_\zeta| \leq k}{\arg\max} \mathsf{DS}(\mathcal{G}_\zeta) \tag{3}$$

### C. Computational Complexity

We next investigate the hardness of evaluating skyline exploratory queries. We start with a verification problem.

*Verification of Explanations*. Given a query $\mathsf{SXQ}^k = (G, \mathcal{M}, v_t, \Phi)$, and a $k$-set of subgraphs $\mathcal{G}$ of $G$, the verification problem is to decide if for all $G_s \in \mathcal{G}$, $G_s$ is a factual or a counterfactual explanation of $\mathcal{M}(v_t, G)$.

**Theorem 1:** *The verification problem for* $\mathsf{SXQ}^k$ *is in P.* □

**Proof sketch:** Given an $\mathsf{SXQ} = (G, \mathcal{M}, v_t, \Phi)$ 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 $G_s \in \mathcal{G}$, if $G_s$ is a factual or a counterfactual explanation of $\mathcal{M}(v_t, G)$. It has been verified that this process can be performed by invoking a polynomial time inference process of $\mathcal{M}$ [34], [38] for $v_t$ over $G_s$ (for testing factual explanation) and $G \setminus G_s$ (for testing counterfactual explanations), respectively. □

While it is tractable to verify explanations, the evaluation of $\mathsf{SXQ}^k$ 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.

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

**Proof sketch:** It suffices to show that $\mathsf{EVAL}(\mathsf{SXQ}^k)$ for an interpretable space $|\zeta| = f(|G|)$ for some polynomial function $f$, and $|\Phi| = 3$. We show this by constructing a polynomial-time reduction from $k$-representative skyline selection problem ($k$-RSP). Given a set $S$ of $d$-dimensional data points, $k$-RSP is to compute a subset $S^*$ of $S$, such that (a) $S^*$ is a set of skyline points, $|S^*| = k$, and (b) $S^*$ maximizes the dominance score. $k$-RSP is NP-hard even for 3-dimensional data points [22].

Given an instance of $k$-RSP with a set $S$ of $n$ 3-dimensional data points, the reduction sets a graph $G$ as a two-level tree with a single root $v_t$, and $n$ distinct edges $(v_i, v_t)$, one for each data point $s_i \in S$. All the nodes in $G$ are assigned the same feature vector and the same label $l$. Duplicate graph $G$ as a training graph $G_T$ and train a vanilla GCN $\mathcal{M}$ with $L$=2, with an output layer a trivial function that sets an output $\mathcal{M}(v_t, G) = l$. As such, any sub-tree rooted at $v_t$ in $G$ is a factual explanatory subgraph $G_\epsilon$ for the output $\mathcal{M}(v_t, G) = l$. For $\Phi$, we fix one of the measures $\phi_1 \in \Phi$ to be "conciseness", hence enforcing the interpretable space to contain only singleton edges, *i.e.,* any explanatory subgraph $G'_\epsilon$ with more than one edges is strictly dominated by a single edge counterpart.

We next *break ties* of all size-1 explanatory subgraphs as follows. For each 3-dimensional data point $s_i \in S$ as a triple $(d_i^1, d_i^2, d_i^3)$, recall that $s$ dominate $s'$ iff $s[i] \geq s'[i]$ for $i \in [1, 3]$, and there is at least one dimension $j$ such that $s[j] > s'[j]$. Assume *w.l.o.g.* that dimension 3 is the "determining" dimension asserting the dominance of data points. Accordingly, we set $\phi_3$ as the determining dimension, and $\phi_2$ an order-preserving encoding of dimensions 1 and 2 that preserve original dominance, for each pair of data points in $S$. This obtains, for each $G_\epsilon$, a 3-D measure $\Phi(G_\epsilon)$ accordingly. We can verify that $s$ dominates $s'$, iff $G_\epsilon = (v_s, v_t)$ dominates $G'_\epsilon = (v'_s, v_t)$. Given that $|\zeta| = |G|$, the above construction is in polynomial time. Consistently, a $k$-representative set $S'$ for $k$-RSP induces an optimal $k$-explanation for $\mathcal{M}(v_t, G)$, and vice versa. The hardness of $\mathsf{EVAL}(\mathsf{SXQ}^k)$ hence follows. □

A straightforward evaluation of $\mathsf{SXQ}^k$ may explore the $L$-hop subgraph $G^L(\{v_t\})$ and initializes the interpretable space $\zeta$ as all *connected* subgraphs in $G^L(\{v_t\})$, by invoking subgraph enumeration algorithms [39], [40]. It then enumerates $n$ size-$k$ subsets of $\zeta$ and finds an optimal $k$-explanation. Although this correctly finds optimal explanations, it is not practical for large $G$, as $n$ alone can be already $2^{deg^L}$ and the Pareto sets to be inspected can be $\binom{n}{k}$. We thus consider "approximate query processing" scheme for $\mathsf{SXQ}^k$, and present efficient algorithms that do not require such enumeration.

## IV. Generating Skyline Explanations

### A. Approximating Skyline Explanations

We introduce our first algorithm, which answers skyline explanatory queries with relative quality guarantee. To this end, we introduce a notion of $\epsilon$-explanation.

**$\epsilon$-explanations**. Given explanatory measures $\Phi$ and an interpretable space $\zeta$, we say that an explanatory subgraph $G_\zeta \in \zeta$ is *$\epsilon$-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 $\mathsf{SXQ}^k = (G, \mathcal{M}, v_t, \Phi)$, $\mathcal{M}$, and $v_t$, an explanation $\mathcal{G}_\epsilon \subseteq \zeta$ is an *$(\zeta, \epsilon)$-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 $G_\zeta \preceq_\epsilon G'_\zeta$.

In other words, a $(\zeta, \epsilon)$-explanation $\mathcal{G}_\epsilon$ approximates a $k$-explanation $\mathcal{G}_\zeta$ as its answer in the interpretable space $\zeta$. 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 *$\epsilon$-Pareto set* [41] of $\zeta$. In multi-objective decision making, $\epsilon$-Pareto sets have been verified as a class of cost-effective, size-bounded approximation for Pareto optimal solutions.

**$(\alpha, \epsilon)$-Approximations**. Given a $k$-skyline query $\mathsf{SXQ}^k$ $(G, \mathcal{M}, v_t, \Phi)$, and an interpretable space $\zeta$ *w.r.t.* $G$, $\mathcal{M}$, and $v_t$, let $\mathcal{G}^*_\zeta$ be the optimal $k$-explanation answer for $\mathsf{SXQ}^k$ in $\zeta$ (see § III-B). We say that an algorithm is an *$(\alpha, \epsilon)$-approximation* for the problem $\mathsf{EVAL}(\mathsf{SXQ}^k)$ *w.r.t.* $\zeta$, if it ensures the following:
- it correctly computes an $(\zeta, \epsilon)$-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* $\mathsf{EVAL}(\mathsf{SXQ}^k)$ w.r.t. $\zeta'$*, where* $\zeta'$ *is the set of explanatory subgraphs verified by the algorithm. The algorithm computes a* $(\zeta', \epsilon)$*-explanation in time* $O(|\zeta'|(\log \frac{r_\Phi}{\epsilon})^{|\Phi|} + |\zeta'|L|G^L(v_t)|)$. □

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 $L$-hop neighbor subgraph of node $v_t$, 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 $r_\Phi$ is often small.

As a constructive proof of Theorem 3, we introduce an approximation algorithm for $\mathsf{EVAL}(\mathsf{SXQ}^k)$.

**Algorithm**. Our first algorithm, denoted as ASX-OP (illustrated as Algorithm 1), takes advantage of a *data locality*

**Algorithm 1** ASX-OP Algorithm

**Input:** a query $\mathsf{SXQ}^k = (G, \mathcal{M}, v_t, \Phi)$; a constant $\epsilon \in [0, 1]$;
**Output:** a $(\zeta', \epsilon)$-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**
4: initializes state $s_0$:=$G^l(v_t)$.
5: **while** $E_l \neq \emptyset$ **do**
6: **for** $e \in E_l$ **do**
7: spawns a state $s$ with candidate $G_s$:=$G^l \backslash \{e\}$;
8: update $\zeta'$ with state $s$ and new transaction $t$;
9: **if** vrfyF($s$)=False & vrfyCF($s$)=False **then**
10: continue;
11: $\mathcal{G}_\epsilon$:=updateSX $(s, G_s, \zeta', \mathcal{G}_\epsilon)$; $E_l$:=$E_l \backslash \{e\}$;
12: **return** $\mathcal{G}_\epsilon$.

property: For a GNN $\mathcal{M}$ with $L$ layers and any node $v_t$ in $G$, its inference only involves the nodes up to $L$-hop neighbors of $v$ via message passing, regardless of how large $G$ is. Hence, it suffices to explore and verify connected subgraphs in $G^L(V_T)$ (see § II). In general, it interacts with three procedures:

(1) a Generator, which initializes and dynamically expands a potential interpretable space $\zeta'$, by generating a sequence of candidate explanatory subgraphs from $G^L(V_T)$;

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

(3) an Updater, that dynamically maintains a current size $k$ $(\zeta', \epsilon)$- explanation $\mathcal{G}_\epsilon$ over verified candidates $\zeta'$, 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), or upon an ad-hoc request at any time from the queryer.

*Auxiliary structures*. ASX-OP coordinates the interaction of the Generator, Verifier, and Updater via a state graph (simply denoted as $\zeta'$). Each node (a "state") $s \in \zeta'$ records a candidate $G_s$ and its local information to be updated and used for evaluating $\mathsf{SXQ}^k$. There is a directed edge (a "transition") $t$ = $(s, s')$ in $\zeta'$ if $G_{s'}$ is obtained by applying a graph editing operator (*e.g.,* edge insertion, edge deletion) to $G_s$. A path $\rho$ in the state graph $\zeta'$ consists of a sequence of transitions 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 $B(s)$ keeps the $\epsilon$-dominance relation between $G_s$ and current candidates in $\zeta'$. Its $\mathsf{DS}(G_s)$ score over $\zeta'$, can be readily counted as the number of "1" entries in $B(s)$.

**"Onion Peeling"**. To reduce unnecessary verification costs, ASX-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 a corresponding initial state $s_0$ that iteratively removes edges from the "outmost" $L$-th hop "inwards" to $v$ (via Generator). This spawns a set of new candidates to be verified (Verifier), and maintained (Updater).

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, $\Phi$, and dominance are well defined. In addition, the process enables early detection and skipping of non-dominating candidates (see "Optimization").

*Outline*. Algorithm ASX-OP dynamically maintains $\zeta'$ as the state graph. It first induces and verifies the $L$-hop neighbor subgraph $G^L(v)$, and initializes state node $s_0$ (w.r.t $G_{s_0}$) with $G^L(v)$ and local information. It then induces $L$-batches of edge sets $E_i$, $i \in [1, L]$, from $G^L(v)$ 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 $\zeta'$ 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 $\mathsf{DS}(s)$ 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 a $\mathsf{SXQ}^2$ query in Figure 2 that requests 2-explanations with a measure space $\Phi$ that requires high $\mathsf{fdl}^+$ and $\mathsf{fdl}^-$. ASX-OP starts the generation of subgraphs within the 2-hop subgraph $s_0$, by peeling single edges at hop 2 of $v_t$, i.e., $e_1$, $e_2$, and $e_3$. This spawns three states $s_1$, $s_2$, and $s_3$ to be verified. It chooses $s_3$, which leads to the new states $s_4$ and $e_5$, in response to the removal of $e_1$ and $e_2$, respectively. It continues to verify $s_4$ and $s_5$. As $s_4$ fails the verification (neither factual nor counterfactual), it continues to verify $s_5$ and obtains a current solution $\{s_3, s_5\}$. □

*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 $\zeta'$, $G_{s'} \preceq_\epsilon G_s$ (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

**Procedure 2** Procedure updateSX

**Input:** state $s$, candidate $G_s$, state graph $\zeta'$, explanation $\mathcal{G}_\epsilon$;
**Output:** updated $(\zeta', \epsilon)$-explanation $\mathcal{G}_\epsilon$;

1: initializes state $s$ with 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**
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** $\mathcal{G}_\epsilon$.

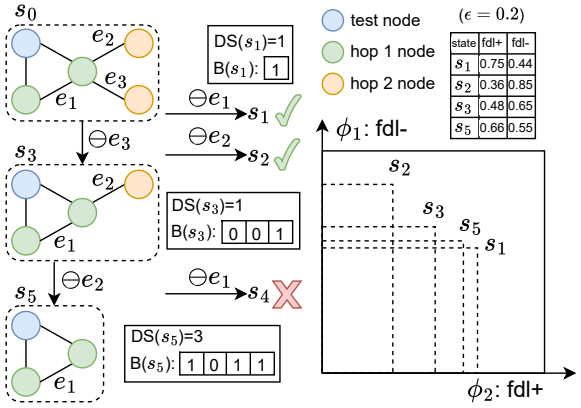



Fig. 2: Illustration of Onion Peeling ($L$=2, $k$=2). $\zeta' = \{s_1, s_2, s_3, s_5\}$. $(\zeta', \epsilon)$-explanation $\mathcal{G}_\epsilon$=$\{s_2, s_5\}$ with DS = 4.

larger size $\mathcal{D}(\mathcal{G}'_\epsilon)$ 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 $\zeta'$. 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. Details are in Appendix.

**Example 5:** Recall Example 4 where the explanation space $\zeta'$ 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 are added to the explanation $\{s_1, s_2\}$. Upon the arrival of $s_3$, since it introduces an improvement less than a factor of $1+\frac{1}{k} = \frac{3}{2}$, updateSX skips $s_3$. As $s_5$ $(1+\epsilon)$-dominates $s_1$ and $s_3$, it replaces $s_1$ with $s_5$ and updates the explanation to be $\{s_2, s_5\}$. □

**Explainability**. Algorithm ASX-OP terminates as it constantly removes edges from $G^L(v_t)$ and verifies a finite number of candidates. We also show its quality guarantee, as stated below.

**Lemma 4:** *Given a constant $\epsilon$,* ASX-OP *correctly computes a $(\zeta', \epsilon)$-explanation of size $k$ defined on the interpretation space $\zeta'$, which contains all verified candidates.* □

**Proof sketch:** We show the above result with a reduction to the multi-objective shortest path problem (MOSP) [42]. Given an 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 $\rho_w$ 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 $\zeta'$ 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 $\Pi_w$ of the above instance of MOSP, for each path $\rho_w \in \Pi$, we set a corresponding path $\rho$ in $\zeta'$ that ends at a state $\rho_s$ , and adds it into $\mathcal{G}_\epsilon$. We can verify that $\Pi_w$ is an $\epsilon$-Pareto set of paths $\Pi_w$ in $G_w$, if and only if $\mathcal{G}_\epsilon$ is an $(\zeta', \epsilon)$-explanation of $\zeta'$. We show that ASX-OP performs a simpler process of the algorithm in [42], which ensures to generate $\mathcal{G}_\epsilon$ as a $(\zeta', \epsilon)$-explanation. □

**Lemma 5:** ASX-OP *computes a $(\zeta', \epsilon)$-explanation $\mathcal{G}_\epsilon$ that ensures* DS$(\mathcal{G}_\epsilon) \geq \frac{1}{4}$DS$(\mathcal{G}^*_\epsilon)$*, where $\mathcal{G}^*_\epsilon$ is the size $k$ $(\zeta', \epsilon)$-explanation with maximum dominance power* 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 $(\zeta', \epsilon)$-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 [43]. 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 [43]. Following the replacement policy consistently, updateSX ensures a $\frac{1}{4}$ approximation ratio for $(\zeta', \epsilon)$-explanations in $\zeta'$. □

**Time Cost**. As Algorithm ASX-OP processes candidate subgraphs in an online fashion from a stream, we analyze its time complexity in an output-sensitive manner in terms of $\zeta'$ at termination. It evaluates $|\zeta'|$ candidate subgraphs. For each candidate $G_s$, the Verifier module (procedures vrfyF and vrfyCF) checks its validity for SXQ via two forward passes of the GNN model $\mathcal{M}$, incurring a cost of $O(L|G^L(v_t)|)$ per candidate, assuming a small number of node features (cf. Lemma 1, [34], [38]). Hence, the total verification cost is $O(|\zeta'|L|G^L(v_t)|)$. Verified candidates are passed to the Updater module (updateSX), which maintains a $(1+\epsilon)$-approximate Pareto set in $\zeta'$ by solving a multi-objective dominance update over the state space. This takes at most $O\left(\prod_{i=1}^{|\Phi|}\left(\left\lfloor \log_{1+\epsilon} \frac{\phi_u}{\phi_l} \right\rfloor + 1\right)\right)$ time per update, where $\phi_l$ and $\phi_u$ are the minimum and maximum values observed for each metric $\phi \in \Phi$. Approximating $\log_{1+\epsilon} \approx \frac{1}{\epsilon}$ for small $\epsilon$, the

total maintenance cost becomes $O\left(|\zeta'| \cdot \left(\frac{\log r_\Phi}{\epsilon}\right)^{|\Phi|}\right)$, where $r_\Phi = \frac{\phi_u}{\phi_l}$. Therefore, the total time cost of ASX-OP is in $O\left(|\zeta'| \cdot \left(\left(\frac{\log r_\Phi}{\epsilon}\right)^{|\Phi|} + L|G^L(v_t)|\right)\right)$.

Putting the above analysis together, Theorem 3 follows.

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

*Edge Prioritization*. ASX-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 (ASX-OP, line 7). It ranks each transaction $t$=$(s, s')$ in $\zeta$ 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.,* $w(t) = \frac{1}{|\Phi|}\sum_{i=1}^{|\Phi|}(\phi_i(s) - \hat{\phi_i}(s'))$. The candidates from spawned states with the 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*. ASX-OP also exploits a *monotonicity property* of measures $\phi \in \Phi$ to early determine the non-$\epsilon$-dominance of an explanatory subgraph. Given $\zeta'$ and a measure $\phi \in \Phi$, we say $\phi$ is *monotonic w.r.t.* a path $\rho$ in $\zeta'$, 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 $\rho$, $\phi_l(G_s) \geq \frac{\phi_u(G_{s'})}{1+\epsilon}$, 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 $\phi$ 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 $\phi$ alone. 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 (estimated) ranges of such measures. Note that the above property is checkable in $O(|\zeta'|)$ time.

### *B. Diversified Skyline Explanations*

Skyline explanation may still contain explanatory subgraphs having highly "similar" or overlapped nodes. This may lead to redundant and biased explanations. In practice, diversified explanations often clarify the model output more comprehensively [44], [45]. We next investigate a diversified scheme for $\mathsf{SXQ}^k$ evaluation, in terms of coverage and the difference between node representations.

**Diversification of Explanatory Query Evaluation**. Given a query $\mathsf{SXQ}^k = (G, \mathcal{M}, v_t, k, \Phi)$, the diversified evaluation of $\mathsf{DivEVAL}(\mathsf{SXQ}^k)$, is to find a $(\zeta', \epsilon)$-explanation $\mathcal{G}_\epsilon$, s.t.

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

where $\mathsf{DivS}(\mathcal{G}_\epsilon)$ is a *diversification function* defined on $\mathcal{G}_\epsilon$, to quantify its overall diversity, and is defined as

$$\mathsf{DivS}(\mathcal{G}_\epsilon) = \alpha \cdot \mathsf{NCS}(\mathcal{G}_\epsilon) + (1-\alpha) \cdot \sum_{G_s, G_{s'} \in \mathcal{G}_\epsilon} \mathsf{CD}(G_s, G_{s'}) \quad (5)$$

here NCS, a *node coverage* measure, aggregates the node coverage of explanatory subgraphs in $\mathcal{G}_\epsilon$:

$$\mathsf{NCS}(\mathcal{G}_\epsilon) = \frac{|\bigcup_{G_s \in \mathcal{G}_\epsilon} V_{G_s}|}{|V_{G^L}|}; \quad (6)$$

and CD, an *accumulated difference* measure, aggregates the node difference between two subgraphs in terms of Cosine distances of their embeddings:

$$\mathsf{CD}(G_s, G_{s'}) = 1 - \frac{\mathbf{x}_{G_s} \cdot \mathbf{x}_{G_{s'}}}{||\mathbf{x}_{G_s}||_2 \cdot ||\mathbf{x}_{G_{s'}}||_2} \quad (7)$$

where $\mathbf{x}_{G_s}$ is the embedding of $G_s$ obtained by node embedding learning such as Node2Vec [46]. The two terms are balanced by a constant $\alpha$.

**Diversification Algorithm**. We next outline our diversified evaluation algorithm, denoted as DSX (pseudo-code shown in Appendix). It follows ASX-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 $\mathsf{DivS}(\mathcal{G}_\epsilon)$. DSX terminates when a first $k$-skyline explanation is found.

*Procedure* updateDSX. For each new candidate $G_s$ (state $s$), updateDSX 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 $\zeta'$, $G_{s'} \preceq_\epsilon G_s$. 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, updateDSX adds it in the $k$-skyline explanation $\mathcal{G}_\epsilon$.

**Quality guarantee**. Adding diversification still permits an approximation scheme with relative quality guarantees, with the same worst-case time cost.

**Lemma 6:** *Given a constant $\epsilon$,* DSX *correctly computes a $(\zeta', \epsilon)$-explanation of size $k$ defined on the interpretation domain $\zeta'$, which contains all verified candidates.* □

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

**Theorem 7:** DSX *computes a $(\zeta', \epsilon)$-explanation $\mathcal{G}_\epsilon$ that ensures* $\mathsf{DivS}(\mathcal{G}_\epsilon) \geq (\frac{1}{2} - \epsilon)\mathsf{DivS}(\mathcal{G}_\epsilon^*)$*, in time $O(|\zeta'|(\log \frac{r_\Phi}{\epsilon})^{|\Phi|} + |\zeta'|L|G^L(v_t)|)$; where $\mathcal{G}_\epsilon^*$ is the size-$k$ $(\zeta', \epsilon)$-explanation over $\zeta'$ with optimal diversity* $\mathsf{DivS}(\mathcal{G}_\epsilon^*)$. □

We show the above result by reducing the diversified evaluation of $\mathsf{SXQ}^k$ to an instance of the Streaming Submodular Maximization problem [47]. The problem maintains a size-$k$ set that optimizes a submodular function over a stream of data objects. We show that DSX adopts a consistent increment policy that ensures a $(\frac{1}{2} - \epsilon)$-approximation as in [47]. We present the detailed proof in Appendix.

## V. Parallel Skyline Explanatory Querying

We next scale our approximate algorithms to query workloads. It is desirable to parallelize the evaluation of a batch

of queries involving a set of targeted nodes $V_T$ to multiple threads. For example, a server may receive a set of SXQ queries requesting fast explanations over a set $V_T$ of newly detected illicit IP addresses from multiple analysts.

A random distribution of queries across threads may lead to redundant verification or "stranglers", due to the redundant computation of overlapped neighbors of the nodes in $V_T$. We thus introduce a parallel algorithm with a workload clustering strategy to maximize the parallelism.

**Parallel Algorithm**. The parallel algorithm, denoted as ParaSX, works with $m$ threads to evaluate a query set $Q$ with $n$ SXQ queries, involving a test node set $V_T$ (pseudocode given in Appendix). (1) *Inter-Batch parallelization*: It first partitions $Q$ into $N$ small batches to be evaluated in parallel, guided by a clustering strategy with a goal that for any pair of target nodes $v_t$ and $v'_t$ from two different batches, their $L$-hop neighbors have minimized overlap (see "Workload Clustering"). (2) *Batch Assignment*. ParaSX then solves a makespan minimization problem to assign the $N$ batches to $m$ threads, with a dynamic scheduling strategy that prioritizes the processing of queries with "influencing" output nodes that have highest common $L$-hop neighbors for those in the same thread. This is to maximize the shared computation for edge peeling and verification. (3) *Inter-query Pipelining*: Each local thread then invokes ASX-OP $(v_t)$ to generate local $k$-explanations for each assigned node $v_t$. All the processing follows its local ranks of highly influential nodes, and automatically skips processing a node, if its information has been updated via peeling operations earlier by previous nodes. This processing strategy is enabled by maintaining a shared data structure that records nodes and their $L$-hop neighboring edges within each thread. The skyline computation can be further parallelized by parallel skyline query methods [48], [49].

**Parallelization Strategies**. ParaSX adopts the two strategies below. (1) Workload Clustering. The workload partitioning is guided by solving an $N$-clustering $\Psi$ of $Q$ that maximizes the accumulated intra-cluster Jaccard similarity (J) [50] of the $L$-hop neighbors of the targeted nodes in each cluster. This induces query workloads in batches with targeted nodes sharing the most common $L$-hop neighbors. The accumulated pairwise intra-similarity of SXQ queries is defined as $\sum_{\mathsf{SXQ}_1^k,\mathsf{SXQ}_2^k \in \Psi(t)} J(N^L(v_t^1), N^L(v_t^2))$, where $N^L(\cdot)$ denotes the $L$-hop neighbor set of a given node, and $v_t^1$ and $v_t^2$ denote the test nodes of $\mathsf{SXQ}_1^k$ and $\mathsf{SXQ}_2^k$, respectively. (2) *Influential Node Prioritization*. The Inter-query pipelining exploits the shared global data structure to dynamically update an influencing score of the nodes. Specifically, each thread begins by applying the $k$-core decomposition algorithm [51] to assign a core number to each node, reflecting the highest $j$ such that the node belongs to the $j$-core. Test nodes are then ranked in descending order of core number, prioritizing more "central" nodes in the thread. This ordering ensures that edge information from structurally important nodes is shared early, maximizing reusability (see more details in Appendix).

TABLE III: Statistics of datasets

| dataset | # nodes | # edges | # node features | # class labels |
|---|---|---|---|---|
| Cora [52] | 2,708 | 10,556 | 1,433 | 7 |
| PubMed [53] | 19,717 | 88,648 | 500 | 3 |
| FacebookPage [54] | 22,470 | 342,004 | 128 | 4 |
| AmazonComputer [55] | 13,752 | 491,722 | 767 | 10 |
| OGBN_arxiv [56] | 169,343 | 1,166,243 | 128 | 40 |
| BA_Shapes [6] | 1,010,000,000 | 6,027,852,000 | 1 | 4 |

## VI. Experimental Study

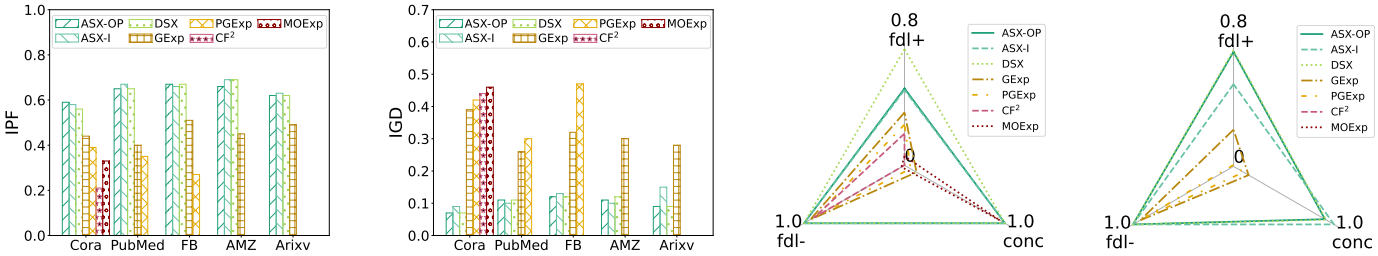
We conduct experiments to evaluate the effectiveness, efficiency, and scalability of our solutions. Our algorithms are implemented in Python 3.10.14 by PyTorch-Geometric framework. All experiments are conducted on a Linux system equipped with AMD Ryzen 9 5950X CPU, an NVIDIA GeForce RTX 3090, and 32 GB of RAM. **Our code and data are made available at [57]**.

### A. Experimental Setup

**Datasets.** Used datasets are summarized in Table III. (1) Cora [52] and PubMed [53] 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 [54], 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) AmazonComputer [55] is a product network. 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 [56] 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. (5) BA_Shapes [6] is a billion-scale synthetic graph, augmented with house-shaped motifs. Each node belongs to one of four classes: top, middle, or bottom (of a house motif), or non-house (not part of any motif).

**GNN Classifiers.** We employ three classes of mainstream GNNs: (1) *Graph convolutional network* (GCN) [29], one of the classic message-passing GNNs; (2) *Graph attention networks* (GAT) [30] leverage attention mechanisms to dynamically weigh the importance of a node's neighbors during inference; and (3) *Graph isomorphism networks* (GIN) [31] with enhanced expressive power up to the Weisfeiler-Lehman graph isomorphism tests.

**GNN Explainers.** We have implemented the following.
(1) Our skyline exploratory query evaluation methods include two approximations ASX-OP (§ IV-A) and the diversification algorithm DSX (§ IV-B). To evaluate the benefit of onion peeling strategy, we also implemented ASX-I, a variant of




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

Fig. 3: Overall effectiveness of our skyline explanation vs. SOTA explainers (GExp, PGExp, CF$^2$, MOExp).

ASX-OP that follows an "edge insertion" strategy to grow candidates up to its $L$-hop neighbor subgraph. (2) Parallel algorithm ParaSX, and its variants ParaSX-N and ParaSX-EIS. ParaSX-N is a naïve parallel algorithm that randomly assigns query loads to each thread without any load balancing strategy. ParaSX-EIS is a parallel algorithm that only adopts asynchronous influential node prioritization with a global data structure and randomly assigns query loads to each thread.

We have employed state-of-the-art GNN explainers for comparison. (1) GExp 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 [6]. (2) PGExp 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 [10]. (3) CF$^2$ optimizes a linear function of weighted factual and counterfactual measures. It learns feature and edge masks, producing effective and simple explanations [8]. (4) MOExp finds Pareto-optimal explanations, striking a balance between "simulatability" (factual) and "counterfactual relevance" [11]. Unlike our methods where users simply set $k$ to bound the output of skyline explanations, GExp, PGExp, and CF$^2$ return only one subgraph, while MOExp has no control of the number of explanations and may return ∼200 subgraphs in our test.

**Evaluation Metrics.** For all datasets and GNNs, we select three common explainability measures ($\Phi$): 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**) [58]–[61]. 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) [62]. 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:

$$\mathsf{nIPF}(\mathcal{G}_A) = \frac{1}{|\mathcal{G}_A| \cdot |\Phi|} \sum_{G \in \mathcal{G}_A} \sum_{\phi \in \Phi} \phi(G) \tag{8}$$

(2) **QI-2**: Inverted Generational Distance (IGD) [58], [63], 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 $\mathcal{G}_\phi^k \in \mathcal{G}$ with explanatory subgraphs having the top-$k$ values in $\phi$. (b) The normalized IGD of an explanation $\mathcal{G}_A$ from an explainer $A$ is defined as:

$$\mathsf{nIGD}(\mathcal{G}_A) = \frac{1}{k \cdot |\Phi|} \sum_{\phi \in \Phi} \sum_{G' \in \mathcal{G}_\phi^k} \min_{G \in \mathcal{G}_A} d(\phi(G), \phi(G')) \tag{9}$$

$d(\cdot)$ is Euclidean distance function following [58], [59].

(3) **QI-3**: Maximum Spread (MS) [58]. 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 $\phi$ is computed as:

$$\mathsf{nMS}(\mathcal{G}_A)^\phi = \frac{\phi(G_\phi^{A*})}{\phi(G_\phi^*)} \tag{10}$$

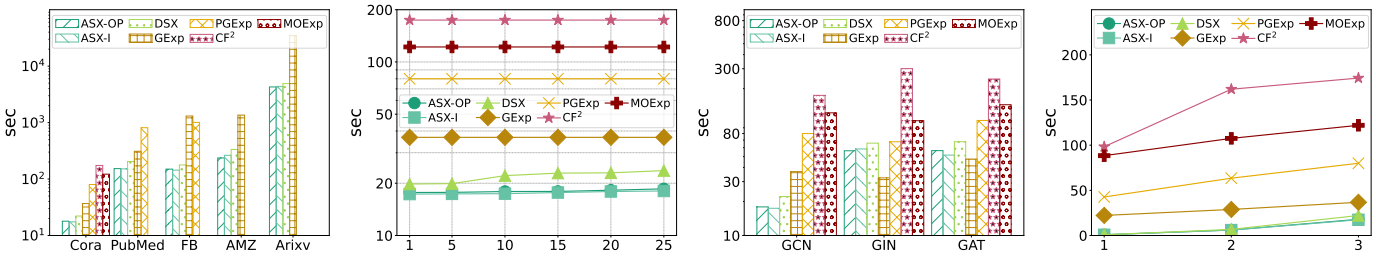
where $G_\phi^*$ is the explanatory subgraph with the best score on $\phi$ in universal set $\mathcal{G}$, and $G_\phi^{A*}$ is $\mathcal{G}_A$'s counterpart on $\phi$.

(4) **Efficiency**: We report the total time of explainers. For learning-based methods, we include learning costs.

### B. Experimental Results

**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 Appendix. (1) In general, ASX-OP outperforms a majority of competitors in multi-objective explainability evaluation metrics. For example, on Cora, ASX-OP outperforms GExp, PGExp, CF$^2$, and MOExp in IPF scores by 1.34, 1.51, 2.81, and 1.79 times, respectively. DSX and ASX-I achieve comparable performance with ASX-OP. (2) We also observe that GExp and PGExp are sensitive as the datasets vary. For example, on FacebookPage, both show a significant change in IPF scores. In contrast, ASX-OP, ASX-I, and DSX consistently achieve top IPF scores over all datasets. This verifies their robustness in generating high-quality explanations over different data sources.




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

Fig. 4: Efficiency and scalability of our skyline explanation vs. SOTA explainers (GExp, PGExp, CF$^2$, MOExp).

*QI-2: IGD Scores*. Figure 3(b) reports the IGD scores (*smaller* is better) of the explanations for GCN-based classification. ASX-OP, ASX-I, and DSX 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 measures. In particular, we find that ASX-OP, ASX-I, and DSX are able to "recover" the top-$k$ global optimal solutions for each individual explainability measures with high hit rates ($\approx$ 90% of the cases).

*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) ASX-OP reports a large range and contributes to explanations that are either close or the exact optimal explanation, for each of the three measures. ASX-I and DSX have comparable performance, yet with larger ranges due to its properly diversified solution. (2) DSX, ASX-OP, and ASX-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 (DSX) 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) ASX-OP, ASX-I, and DSX outperform all (learning-based) GNN explanations. ASX-OP (resp. DSX) on average outperforms GExp, PGExp, CF$^2$, and MOExp by 2.05, 4.46, 9.73, and 6.81 (resp. 1.62, 3.61, 7.88, and 5.51) times, respectively. Moreover, ASX-OP supersedes GExp and PGExp 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.

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

(3) DSX does not incur significant overhead despite of diversified evaluation, because the benefit of prioritization carries over to the diversified search. The incremental maintenance of $k$-explanations further reduces unnecessary verification.

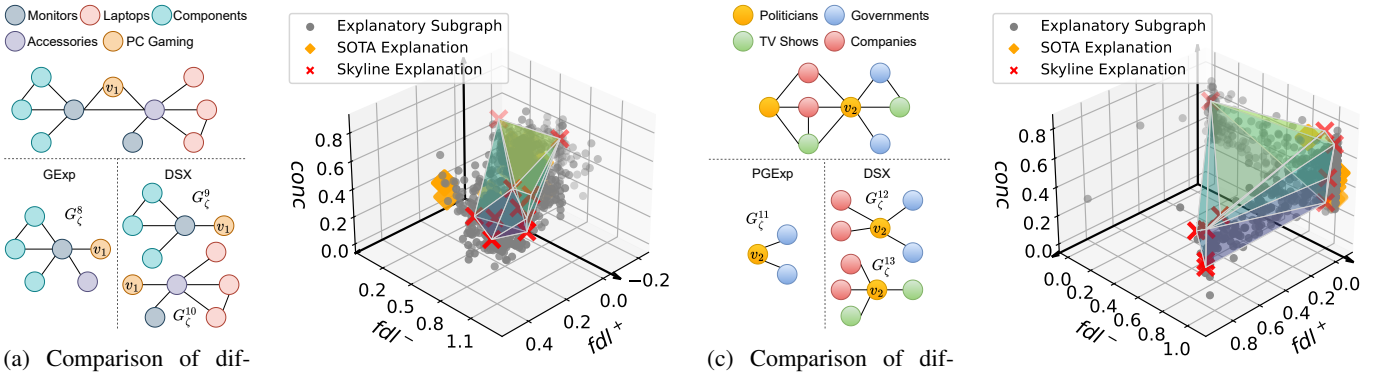
(4) ASX-I may outperform ASX-OP for cases when the test nodes have "skewed" edge distribution in $L$-hop subgraphs, i.e., less direct neighbors but more "remote" neighbors, which favors the edge growth strategy of ASX-I.

**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 results about effectiveness by varying factors are in Appendix.

*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 ASX-OP, ASX-I, and DSX (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. DSX is relatively more sensitive to $k$ due to additional computation of cosine distances (§IV-B), yet remains 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 $L$*. (1) 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. (2) 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.

*Parallelization*. Setting $\mathcal{M}$ as GCN-based classifier with 2 layers, we vary the number of threads $m$ from 2 to 8 for OGBN_arxiv and 8 to 16 for BA_Shapes, respectively. We show the makespan of processing a query workload of 1000 SXQ for OGBN_arxiv, and 2000 SXQ for BA_Shapes, respectively. As show in Figure 6 (with the dashed horizontal line showing the cost of single-thread processing ASX-OP), our skyline explanation scheme can be effectively parallelized to



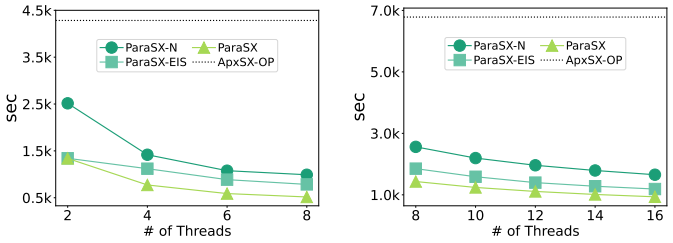

(a) Comparison of different explanations on AmazonComputer.

(b) Visualization of Skyline Explanation (AmazonComputer)

(c) Comparison of different explanations on FacebookPage.

(d) Visualization of Skyline Explanation (FacebookPage)

Fig. 5: Qualitative analysis of our skyline explanation and SOTA explainers (GExp, PGExp, CF$^2$, MOExp).




(a) OGBN_arxiv

(b) BA_Shapes

Fig. 6: Scaling to million-scale and billion-scale graphs.

million-scale and billion-scale graphs, and scales well as more threads are used. ParaSX improves the efficiency of ParaSX-N and ParaSX-EIS by 1.8 and 1.4 times on average, due to load balancing and scheduling strategies.

**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.,* GExp) return explanations (see $G_\zeta^8$ in Figure 5(a)) that simply reveals a fact that $v_1$ is co-purchased with "Components", a "one-sided" interpretation. In contrast, DSX 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 patterns bridging $v_1$ with not only "Components" for "Monitors" (by $G_\zeta^9$), but also to "Accessories" of "Laptops". Indeed, the former indicates gamers who prefer gaming PC with high-refresh rate monitors; and the latter indicates that $v_1$ is a gaming laptop in need of frequent maintenance with laptop accessories. This showcase that DSX provides 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., GExp, PGExp, 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 (Linear Combination).* Our second case compares DSX and PGExp (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 PGExp generates small factual subgraphs, such as $G_\zeta^{11}$ for a test node $v_2$, yet relatively less informative. DSX 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.

## VII. Conclusion

We have proposed a class of skyline explanations that simultaneously optimize multiple explainability measures for interpreting the output of graph neural networks. We have shown that the generation of skyline explanations (as an explanatory query processing problem) is nontrivial even for polynomially bounded input space and three measures. We have introduced feasible algorithms, sequential and parallel, for the skyline explanatory query processing, 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.

## REFERENCES


[1] Y. Wang, Z. Li, and A. Barati Farimani, *Graph neural networks for molecules*. Springer International Publishing, 2023, pp. 21–66.

[2] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[3] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, p. 1082–1090.

[4] L. Wei, H. Zhao, Z. He, and Q. Yao, "Neural architecture search for gnn-based graph classification," *ACM Trans. Inf. Syst.*, vol. 42, no. 1, 2023.

[5] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5782–5799, 2023.

[6] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[7] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, "Cf-gnnexplainer: Counterfactual explanations for graph neural networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022, pp. 4499–4511.

[8] J. Tan, S. Geng, Z. Fu, Y. Ge, S. Xu, Y. Li, and Y. Zhang, "Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning," in *ACM web conference (WWW)*, 2022, pp. 1018–1027.

[9] Y. Zhang, Q. Liu, G. Wang, W. K. Cheung, and L. Liu, "Gear: Learning graph neural network explainer via adjusting gradients," *Knowledge-Based Systems*, vol. 302, p. 112368, 2024.

[10] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[11] Y. Liu, C. Chen, Y. Liu, X. Zhang, and S. Xie, "Multi-objective explanations of gnn predictions," in *IEEE International Conference on Data Mining (ICDM)*, 2021, pp. 409–418.

[12] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *International conference on machine learning (ICLR)*, 2021.

[13] Y. Elmougy and L. Liu, "Demystifying fraudulent transactions and illicit nodes in the bitcoin network for financial forensics," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, p. 3979–3990.

[14] T. Chen, D. Qiu, Y. Wu, A. Khan, X. Ke, and Y. Gao, "View-based explanations for graph neural networks," *Proc. ACM Manag. Data*, vol. 2, no. 1, pp. 40:1–40:27, 2024.

[15] L. Cheng, F. Zhu, Y. Wang, R. Liang, and H. Liu, "Evolve path tracer: Early detection of malicious addresses in cryptocurrency," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, p. 3889–3900.

[16] C. Bellei, M. Xu, R. Phillips, T. Robinson, M. Weber, T. Kaler, C. E. Leiserson, Arvind, and J. Chen, "The shape of money laundering: Subgraph representation learning on the blockchain with the elliptic2 dataset," in *KDD Workshop on Machine Learning in Finance*, 2024.

[17] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, pp. 853–862, 2010.

[18] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems," *Structural optimization*, vol. 14, pp. 63–69, 1997.

[19] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *International Conference on Data Engineering (ICDE)*, 2001, pp. 421–430.

[20] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *ACM SIGMOD International Conference on Management of Data*, 2003, p. 467–478.

[21] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *ACM SIGMOD International Conference on Management of Data*, 2006, p. 503–514.

[22] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: The k most representative skyline operator," in *International Conference on Data Engineering (ICDE)*, 2006, pp. 86–95.

[23] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, 1975.

[24] S. Sharma and V. Kumar, "A comprehensive review on multi-objective optimization techniques: Past, present and future," *Archives of Computational Methods in Engineering*, vol. 29, no. 7, pp. 5605–5633, 2022.

[25] S. Pushpakom, F. Iorio, P. A. Eyers, K. J. Escott, S. Hopper, A. Wells, A. Doig, T. Guilliams, J. Latimer, C. McNamee, A. Norris, P. Sanseau, D. Cavalla, and M. Pirmohamed, "Drug repurposing: progress, challenges and recommendations," *Nature reviews Drug discovery*, vol. 18, no. 1, pp. 41–58, 2019.

[26] H. He, Y. Ji, and H. H. Huang, "Illuminati: Towards explaining graph neural networks for cybersecurity analysis," in *IEEE European Symposium on Security and Privacy*, 2022, pp. 74–89.

[27] S. Ouyang, Q. Bai, H. Feng, and B. Hu, "Bitcoin money laundering detection via subgraph contrastive learning," *Entropy*, vol. 26, no. 3, 2024.

[28] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference (WWW)*, 2019, p. 417–426.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[30] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[31] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations (ICLR)*, 2019.

[32] M. S. Schlichtkrull, N. De Cao, and I. Titov, "Interpreting graph neural networks for nlp with differentiable edge masking," in *International Conference on Learning Representations (ICLR)*, 2020.

[33] S. Zhang, Y. Liu, N. Shah, and Y. Sun, "Gstarx: Explaining graph neural networks with structure-aware cooperative games," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[34] D. Qiu, M. Wang, A. Khan, and Y. Wu, "Generating robust counterfactual witnesses for graph neural networks," in *IEEE International Conference on Data Engineering (ICDE)*, 2024, pp. 3351–3363.

[35] J. Du, S. Wang, H. Miao, and J. Zhang, "Multi-channel pooling graph neural networks." in *IJCAI*, 2021, pp. 1442–1448.

[36] R. Qiu, X. Jiang, H. Lai, H. Miao, X. Chu, J. Zhao, and Y. Wang, "Efficient graph continual learning via lightweight graph neural tangent kernels-based dataset distillation." in *ICML*, 2025.

[37] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[38] M. Chen, Z. Wei, B. Ding, Y. Li, Y. Yuan, X. Du, and J.-R. Wen, "Scalable graph neural networks via bidirectional propagation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[39] S. Karakashian, B. Y. Choueiry, and S. G. Hartke, "An algorithm for generating all connected subgraphs with k vertices of a graph," *Lincoln, NE*, vol. 10, no. 2505515.2505560, 2013.

[40] Z. Yang, L. Lai, X. Lin, K. Hao, and W. Zhang, "Huge: An efficient and scalable subgraph enumeration system," in *International Conference on Management of Data (SIGMOD)*, 2021.

[41] O. Schütze and C. Hernández, "Computing $\epsilon$-(approximate) pareto fronts," in *Archiving Strategies for Evolutionary Multi-objective Optimization Algorithms*, 2021, pp. 41–66.

[42] G. Tsaggouris and C. Zaroliagis, "Multiobjective optimization: Improved fptas for shortest paths and non-linear objectives with applications," *Theory of Computing Systems*, vol. 45, no. 1, pp. 162–186, 2009.

[43] G. Ausiello, N. Boria, A. Giannakos, G. Lucarelli, and V. T. Paschos, "Online maximum k-coverage," *Discrete Applied Mathematics*, vol. 160, no. 13, pp. 1901–1913, 2012.

[44] Z. Gong, P. Zhong, and W. Hu, "Diversity in machine learning," *Ieee Access*, vol. 7, pp. 64 323–64 350, 2019.

[45] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 2020, pp. 607–617.

[46] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[47] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, "Streaming submodular maximization: massive data summarization on the fly," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, p. 671–680.
[48] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, and A. El Abbadi, "Parallelizing skyline queries for scalable distribution," in *International conference on extending database technology*. Springer, 2006, pp. 112–130.
[49] Y. Park, J.-K. Min, and K. Shim, "Parallel computation of skyline and reverse skyline queries using mapreduce," *Proceedings of the VLDB Endowment*, vol. 6, no. 14, pp. 2002–2013, 2013.
[50] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*, 2nd ed. USA: Cambridge University Press, 2014.
[51] V. Batagelj and M. Zaveršnik, "Fast algorithms for determining (generalized) core groups in social networks," *Advances in Data Analysis and Classification*, vol. 5, no. 2, pp. 129–145, 2011.
[52] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, pp. 127–163, 2000.
[53] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.
[54] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *Journal of Complex Networks*, vol. 9, no. 2, 2021.
[55] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *Relational Representation Learning Workshop @ NeurIPS*, 2018.
[56] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," in *Advances in neural information processing systems (NeurIPS)*, 2020.
[57] "Code and datasets," 2024, *https://github.com/DazhuoQ/SkyExp*.
[58] M. Li and X. Yao, "Quality evaluation of solution sets in multiobjective optimisation: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–38, 2019.
[59] K. Xue, J. Xu, L. Yuan, M. Li, C. Qian, Z. Zhang, and Y. Yu, "Multi-agent dynamic algorithm configuration," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
[60] X. Cai, Y. Xiao, M. Li, H. Hu, H. Ishibuchi, and X. Li, "A grid-based inverted generational distance for multi/many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 21–34, 2021.
[61] A. Morales-Hernández, I. Van Nieuwenhuyse, and S. Rojas Gonzalez, "A survey on multi-objective hyperparameter optimization algorithms for machine learning," *Artif. Intell. Rev.*, vol. 56, no. 8, p. 8043–8093, 2022.
[62] W. M. Carlyle, J. W. Fowler, E. S. Gel, and B. Kim, "Quantitative comparison of approximate solution sets for bi-criteria optimization problems," *Decision Sciences*, vol. 34, no. 1, pp. 63–82, 2003.
[63] C. A. Coello Coello and M. Reyes Sierra, "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*, 2004, pp. 688–697.

## APPENDIX

### A. Details of Algorithms and Procedures

**Procedure** updateSX**: Update Dominance Relation.** We introduce the details of Procedure updateSX; specifically, we describe how the (1+$\epsilon$) dominance relations of exploratory subgraphs are dynamically maintained. With the stream of upcoming states, i.e., explanatory subgraphs, we maintain a lattice structure for the dominance relationships. Specifically, we first compare each newly arrived state $s$ with the current skylines, and incrementally update the dominance lattice by conducting one of the following steps:

- If certain skylines (1+$\epsilon$)-dominate it, it inserts a directed edge from each dominating skylines to the new state; the checking of (1+$\epsilon$)-dominance takes constant time for small measure space involving several explainability measures;
- If the new state dominates certain skylines, we connect directed edges from the new state to each dominated skylines; Meanwhile, we reconnect the dominant states of these skylines to the new state;
- If the new state is not dominated by any skyline, then we obtain its $\mathcal{D}(s)$ using the bitvector $B(s)$.

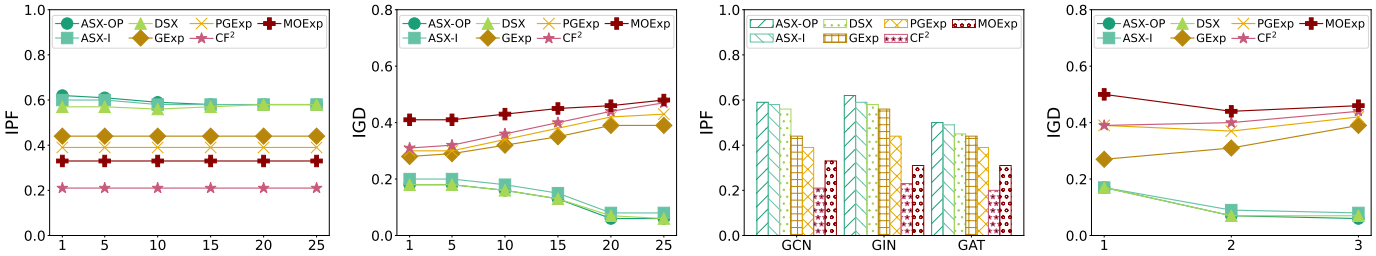
Next, we identify the skyline $\overline{s}$ with the smallest $\mathcal{D}(\overline{s})$, replace $\overline{s}$ with $s$, only when such replacement makes the $\mathcal{D}$ increased by a factor of $\frac{1}{k}$. If swapping is conducted, the new skyline, i.e. $s$, connects directed edges to the dominating 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 [43].

**Algorithm** ASX-I**: Alternative "Edge Growing" Strategy.** As the end user may want early termination and obtain a compact, smaller-sized explanations, we also outline a variant of ASX-OP, denoted as ASX-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. ASX-I remains to be an $(\frac{1}{4}, \epsilon)$-approximation for EVAL(SXQ$^k$), and does not incur additional time cost compared with ASX-OP.

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. ASX-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 ASX-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 ASX-I remains the same as ASX-OP, since the interpretation domain $\zeta'$ and the update strategy are the same.

**Diversification Algorithm.** The detailed pseudo-codes of diversification algorithm DSX are shown as Algorithm 3 and Procedure 3. Please find the descriptions of DSX in § IV-B.

*Approximability and Time cost of* DSX. We can verify that NCS$(\cdot)$ and CD$(\cdot)$ are submodular functions. Consider procedure updateDSX 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 diversified evaluation of SXQ$^k$ to an instance of the Streaming Submodular Maximization problem [47]. The problem maintains a size-$k$ set that optimizes a submodular function over a stream of data objects. The diversification function DivS is submodular, as it combines two submodular components: NCS and the summation over CD. Specifically,




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

Fig. 7: Impact of factors on the effectiveness of explanations

**Algorithm 3** DSX Algorithm

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

set $\mathcal{G}_\epsilon$:=$\emptyset$;
identify edges for each hop: $\mathcal{E}$:=$\{E_L, E_{L-1}, \ldots, E_1\}$;
**for** $l = L$ to 1 **do**
  initializes state $s_0$:=$G^l(v_t)$
  **while** $E_l \neq \emptyset$ **do**
    **for** $e \in E_l$ **do**
      spawns a state $s$ with candidate $G_s$:=$G^l \backslash \{e\}$;
      update $\zeta'$ with state $s$ and new transaction $t$;
      **if** $\mathsf{vrfyF}(s)$=False & $\mathsf{vrfyCF}(s)$=False **then**
        continue;
      $\mathcal{G}_\epsilon$:=$\mathsf{updateDSX}$ $(s, G_s, \zeta', \mathcal{G}_\epsilon)$; $E_l$=$E_l \backslash \{e\}$;
**return** $\mathcal{G}_\epsilon$.

**Procedure** updateDSX

**Input:** a state $s$, candidate $G_s$, state graph $\zeta'$; explanation $\mathcal{G}_\epsilon$;
**Output:** Updated $(\zeta', \epsilon)$-explanation $\mathcal{G}_\epsilon$;

initializes state $s$ with structure $\Phi(G_s)$:=$\emptyset$;
evaluates $\Phi(G_s)$;
incrementally determines $(1+\epsilon)$-dominance of $G_s$;
**if** $\{G_s\}$ is a new skyline explanation **then**
  $\Delta(s|\mathcal{G}_\epsilon) := \mathsf{DivS}(\mathcal{G}_\epsilon \cup \{G_s\}) - \mathsf{DivS}(\mathcal{G}_\epsilon)$
  **if** $|\mathcal{G}_\epsilon| < k$ **and** $\Delta(s|\mathcal{G}_\epsilon) \geq \frac{(1+\epsilon)/2 - \mathsf{DivS}(\mathcal{G}_\epsilon)}{k - |\mathcal{G}_\epsilon|}$ **then**
    $\mathcal{G}_\epsilon := \mathcal{G}_\epsilon \cup \{G_s\}$;
**return** $\mathcal{G}_\epsilon$.

**Algorithm 4** ParaSX Algorithm

**Input:** a set of queries $Q = \{\mathsf{SXQ}_1^k, \mathsf{SXQ}_2^k, \cdots, \mathsf{SXQ}_n^k\}$; the number of threads $m$;
**Output:** a set of $(\zeta', \epsilon)$-explanations $\mathcal{G}_Q$ for $Q$.

$\mathcal{G}_Q := \emptyset$; $V_T := \bigcup_{i=1}^n \mathsf{SXQ}_i^k.v_t$; node partition $P := \emptyset$;
extract $L$-hop neighbors of $V_T$: $G_{V_T}$; globally share $G_{V_T}$, $\mathcal{M}$, $\Phi$;
$P := \mathsf{CP}(V_T, G_{V_T})$;
**for** $i = 1$ to $m$ **do**
  order $p_i$ by computing node influential scores.
  initialize edge information sharing table $eist^i := \emptyset$;
  **for** $v_t \in p_i$ **do**
    ${\mathcal{G}_Q}^i[v_t] := \mathsf{ASX\text{-}OP}\ (v_t)$
    update $eist^i$ with newly visited edges;
$\mathcal{G}_Q := \bigcup_{i=1}^m {\mathcal{G}_Q}^i$;
**return** $\mathcal{G}_Q$.

**Procedure** CP

**Input:** a set of test nodes $V_T$; the subgraph of $V_T$: $G_{V_T}$;
**Output:** a partition $P$ of $V_T$.

generate MinHash signatures for $V_T$;
indexing signatures by Locality-Sensitive Hashing;
partition $V_T$ into $m$ subsets based on LSH indexing: $P := \{p_1, p_2, \cdots, p_m\}$;
Balance the sizes of the partitions;
**return** $P$.

NCS, representing node coverage, is a submodular function bounded within $[0, N]$, with each incremental gain diminishing as more nodes are covered. In parallel, CD denotes the cosine distance between two embeddings, which is bounded in $[0, 1]$. Since pairwise cosine distance is a non-negative and bounded measure, the summation over CD across a set remains submodular. Consequently, their combination in DivS preserves submodularity. DSX 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 [47]. Since DSX follows the same process as ASX-OP but only differs in replacement policy with same time cost, the cost of DSX is also $O(|\zeta'|(\log \frac{r_\Phi}{\epsilon})^{|\Phi|} + |\zeta'|L|G^L(v_t)|)$.

**Parallel Algorithms**. We first present the details of the baseline parallel algorithms (ParaSX-N and ParaSX-EIS). They differ in modeling and minimizing the makespan of skyline explanatory query processing, i.e., the total elapsed (wall-clock) time from the start of the first $\mathsf{SXQ}^k$ to the completion of the last $\mathsf{SXQ}^k$ in a batch. Our parallel algorithms aim to tackle the common makespan minimization problem. Given a batch of $\mathsf{SXQ}^k$ queries $Q = \{\mathsf{SXQ}_1^k, \mathsf{SXQ}_2^k, \cdots, \mathsf{SXQ}_n^k\}$ and a set of threads running in parallel: $T = \{t_1, t_2, \cdots, t_m\}$, the objective is to assign each thread $t_i$ a subset of query workload $Q_i \subseteq Q$, such that the maximum time of any thread takes to

complete its subset of queries (or its makespan) is minimized. More formally, it computes an assignment $\Psi : Q \rightarrow T$ to minimize

$$\max_{t \in T} \sum_{\mathsf{SXQ}^k \in \Psi(t)} c(\mathsf{SXQ}^k)$$

where $c(\cdot)$ is the (estimated) execution time of a given $\mathsf{SXQ}^k$.

*A Naïve solution.* An intuitive approach to minimize the $\mathsf{SXQ}^k$ makespan is randomly splitting $Q$ into $\frac{n}{m}$ subsets and assign each of them to one thread. Within each thread, we can select one skyline generation algorithm of interest: ASX-OP, ASX-I, or DSX, then invoke the algorithm to iteratively generate the skyline explanation for each query. We refer to this baseline approach as ParaSX-N.

*An improved approach*. We next introduce an improved approach, denoted as ParaSX-EIS. The partition strategy of the queries is the same as ParaSX-N–the batch of queries is randomly split and assigned. It differs from ParaSX-N in using a shared edge table to avoid unnecessary computation for common edges in L-hop neighbors of the test nodes; and adopts a node prioritization strategy to dynamically process influencing nodes.

(1) Edge Information Sharing. Considering a subset of skyline explanatory queries assigned to a thread, the computation cost boils down to the computation of the corresponding subset of test nodes. We observe that test nodes often share common neighbors. As noted in Section IV-A, previous optimization techniques compute marginal gains per edge for each individual test node, resulting in redundant computations across overlapping neighborhoods. To mitigate this, we propose an approach that maintains a per-thread lookup set to track visited edges across all test nodes assigned to the same thread.

(2) Influential Node Prioritization. Firstly, since edge information is computed with respect to individual test nodes, any shared edge is attributed to the first node that processes it. To ensure this shared information is representative of other test nodes within the same thread, it is important to prioritize nodes that are more influential or centrally located among the assigned subset. To this end, we propose an influential node prioritization strategy that ranks test nodes within each thread, ensuring that nodes with higher centrality are processed earlier. Each thread begins by applying the $k$-core decomposition algorithm [51] to assign a core number to each node, reflecting the highest $j$ such that the node belongs to the $j$-core. Test nodes are then ranked in descending order of core number, prioritizing more "central" nodes in the thread. This ordering ensures that edge information from structurally important nodes is shared early, maximizing reuse.

Bringing these strategies into the algorithm at runtime, we first perform node prioritization (as described in § IV-A), i.e., the most influential node in the thread. For each edge in its neighborhood, we compute the marginal gain, determine the deletion direction using the "onion-peeling" strategy, and store the resulting edge scores in the lookup set. For subsequent test nodes, any edge already recorded in the lookup set is skipped, and its previously computed marginal gain is directly reused. Only unseen edges are processed and added to the set, thereby reducing redundant computation.

**Algorithm** ParaSX**.** Recall our ultimately proposed parallel algorithm ParaSX in Section V. The key distinction between ParaSX and its two baseline variants (ParaSX-N and ParaSX-EIS) lies in its novel partitioning strategy, which enables more effective node clustering and promotes greater overlap in shared neighborhoods among nodes within each thread.

As illustrated in Algorithm 4, we begin by extracting the $L$-hop neighbors of the query nodes in $V_T$, since GNNs typically rely on at most $L$-hop neighborhood information for each node. Additionally, we globally share the extracted subgraph, the GNN model, and the set of explainability measures (line 2). Next, we perform the workload partitioning using procedure CP (line 3), which clusters the query nodes and assigns each partition to a separate thread. Within each thread, we first perform influential node prioritization (lines 4–5). Then, following the resulting prioritization order, we process each node and update the shared edge information table to reuse previously computed results (lines 6–9). Finally, we aggregate the skyline explanations computed by all threads and return the consolidated results for the entire query batch (lines 10–11).

**Example 6:** As illustrated in Figure 8, six SXQ queries have been issued with specified output nodes $v_1$ to $v_6$. Following the intra-similarity analysis (procedure CP), the query workload is partitioned into three clusters $\{v_1, v_3\}$, $\{v_4, v_5\}$ and $\{v_2, v_6\}$ (the latter two are not shown), by maximizing the intra-cluster Jaccard similarity of their 2-hop neighbors. The nodes in the same cluster are sorted by their core numbers by adapting $k$-core decomposition algorithm [51]. In this case, $v_1$ is scheduled to be processed first, followed by $v_3$. During the processing, all threads consistently invoke ASX-OP, yet share the edge information asynchronously, via the shared edge information table. For example, once the edges in the neighbors of $v_1$ are verified, its auxiliary information is shared for fast local computation without recomputation. For instance, $e_{17}$ is skipped for $v_3$ since it has $e_{17}$ in its neighbor graphs. □

**Parallel Time Cost**. The time cost of ParaSX consists of two parts: (1) The time cost of the cluster-based partitioning, which is: $O\left(|V_T|\left(\max(G^L(v)) + \log m\right)\right)$, where $|V_T|$ is the number of test nodes, $G^L(v)$ denotes the $L$-hop neighborhood of node $v$, and $m$ is the number of threads. (2) The time complexity of the skyline explanation computation, which is: $O\left(\frac{|V_T|}{m}|\zeta'|\left(\left(\log \frac{r_\Phi}{\epsilon}\right)^{|\Phi|} + L|G^L(V_T)|\right)\right)$, where $|\zeta'|$ is the number of candidate explanations, $|\Phi|$ is the feature dimension, $r_\Phi$ is the feature range, and $|G^L(V_T)|$ is the average $L$-hop neighborhood size across all test nodes. Combining the two, the total parallel time complexity for ParaSX is:

$$O\left(|V_T|\left(\max(G^L(v)) + \log m\right) + \frac{|V_T|}{m}|\zeta'|\left(\left(\log \frac{r_\Phi}{\epsilon}\right)^{|\Phi|} + L|G^L(V_T)|\right)\right)$$

The main efficiency gain comes from the term $G^L(V_T)$: since

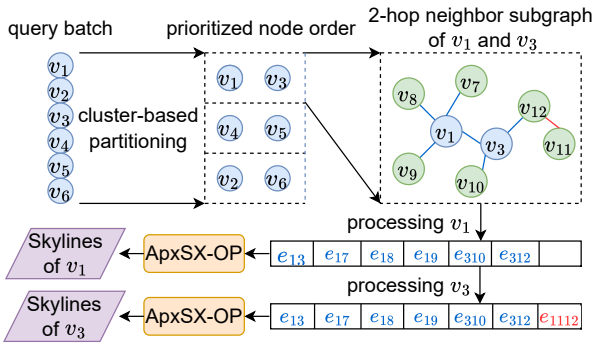



Fig. 8: Running Example for ParaSX. The blue edges are computed when processing $v_1$; The red edge is computed when processing $v_3$. When a query batch containing six nodes arrives, ParaSX first partitions the queries into threads by maximizing intra-cluster Jaccard similarity. Within each thread, the queries are then ordered using the $k$-core algorithm. Consider a thread where the query order is $v_1$ followed by $v_3$. ASX-OP initializes the shared edge information table by computing the edge scores (depicted in blue) from the neighborhood graph of $v_1$. During the processing of $v_3$, only one additional edge—$e_{1112}$ (shown in red)—requires computation, as the remaining edges are shared with $v_1$ and can directly reuse the values stored in the shared table.

each edge is computed only once and shared across relevant nodes, the overall complexity is proportional to the size of the $L$-hop neighborhood of the entire query set $V_T$. In contrast, without edge information sharing, the total cost would scale with the sum of the $L$-hop neighborhoods of individual nodes, i.e., $\sum_{v_t \in V_T} G^L(v_t)$.

### B. Additional Experimental Results

We present additional experimental study and case analysis.

**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 ASX-OP, ASX-I, and DSX (which are dependent on $k$) in Figures 7(a) and 7(b). As $k$ becomes larger, all achieve better scores and can 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

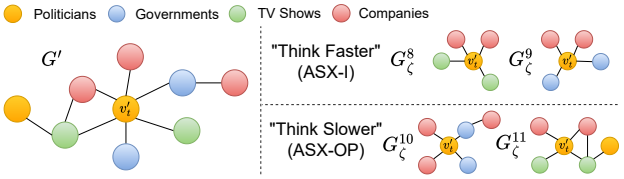



Fig. 9: Different edge explore strategies show different preferences for returned skyline explanation.

results. Meanwhile, even with $k$=1, our methods outperform other competitors for both IPF and IGD.

*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 7(c), ASX-OP, ASX-I, and DSX 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 7(d) shows that ASX-OP, ASX-I, and DSX 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: Edge Growing v.s. Onion-Peeling.** In this case study, we compare the application scenarios of ASX-OP and ASX-I, as shown in Figure 9. We observe that ASX-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 [11]. ASX-I, on the other hand, due to its containment of edges closer to targeted ones via edge insertions/growths, 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 [11]. For example, in answering "why" node $v_t'$ is recognized as a "Politician" site, $G_\zeta^8$ and $G_\zeta^9$ are among skyline explanations from ASX-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. $G_\zeta^{10}$ and $G_\zeta^{11}$, which are reported by ASX-OP in its early stage via "onion-peeling", are relatively larger counterfactual explanations that suggest interaction patterns with more complex structures. For example, $G_\zeta^{11}$ 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.