Deep Skyline Community Search

Minglang Xie[1,2], Jianye Yang[1,3,*], Wenjie Zhang[2], Shiyu Yang[1], Xuemin Lin[4]
[1]*Cyberspace Institute of Advanced Technology, Guangzhou University*, Guangzhou, China
[2]*School of Computer Science, University of New South Wales*, Sydney, Australia
[3]*Department of New Networks, PengCheng Laboratory*, Shenzhen, China
[4]*Antai College of Economics and Management, Shanghai Jiao Tong University*, Shanghai, China
{minglang.xie,wenjie.zhang}@unsw.edu.au,{jyyang,syyang}@gzhu.edu.cn,xuemin.lin@gmail.com

ABSTRACT

Community search has attracted significant research attention over the past decades, which aims to find structurally cohesive subgraphs in a given graph. Recently, the skyline community has been investigated on multi-valued networks where each node is associated with a set of attributes. However, this problem is very challenging, as the computational complexity is exponential to the number of attributes d. As a result, the state-of-the-art algorithm can hardly process datasets with $d \geq 5$. To overcome this challenge, we resort to deep learning techniques and propose a novel framework, called **Deep** **S**kyline **C**ommunity **S**earch (**DeepSCS**). DeepSCS is designed to search skyline communities for given query nodes through a two-phase approach, namely offline pre-training and online search. In the offline pre-training phase, graph convolutional networks (GCN) are adapted to efficiently process high-dimensional data and complex network dependencies by combining the attribute dominance loss and the link loss. In the online search phase, the skyline community score is computed on the basis of learned representations where an expected score gain function based community search algorithm is developed. We conduct extensive experiments on real-world networks. The experimental results show that, DeepSCS can achieve up to 3 orders of magnitude efficiency improvement and competitive accuracy compared to the existing method.

1 INTRODUCTION

Many real-world networks such as social networks, citation graphs, collaboration networks, biological networks can be modeled as multi-valued networks, in which a node with multiple numerical attributes represents an entity of the networks, and an edge represents a link between nodes in a defined relationship of the networks [4][20], providing a rich framework for modeling complex relationships. For example, in the real estate network, each property listing has several numerical attributes, including the price, square footage, number of bedrooms, number of bathrooms, year built, and walkability score, etc. Such network is typically modeled as a multi-valued network where each node is associated with d ($d \geq 1$) numerical attributes. These multi-valued networks also contain community structures, as general networks (i.e., non-attribute networks). Finding the communities in a network is a fundamental problem in network science.

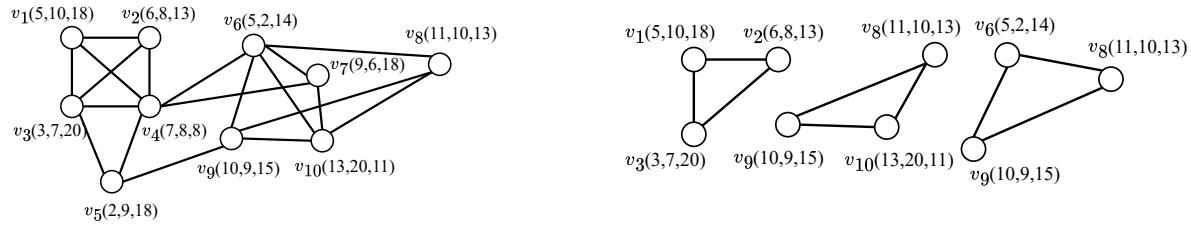
In the past two decades, a query-dependent community discovery problem called Community Search (CS) [10] has attracted significant attention due to its widespread application [8][9][14][21][35]. This problem focuses on identifying densely connected subgraphs given a specific query, which can reveal meaningful structures and relationships within multi-valued networks.

Among existing solutions, the algorithms either only focus on the structural cohesion of the community or just consider one numerical attribute of the nodes. To address this challenge, a novel community model called the skyline community model [9] was introduced. The skyline community is a community method based on the concepts of k-core [8][30][31] and skyline [7][24][25][26]. The skyline community has many applications, particularly in scenarios where multiple criteria must be optimized simultaneously. For example, on social media platforms like Twitter or Facebook, a photography enthusiast can discover influential photography groups that align with specific preferences such as user activity, engagement levels, content preferences, geographic location, and demographic information, etc. Skyline community can provide users with desired communities that meet multiple criteria simultaneously. Therefore, from the skyline communities, the photography enthusiast can find the desired communities to communicate and make appointments to take photos with each other. The application of skyline communities in such multi-valued networks demonstrates the importance of considering both structural cohesiveness constraints and the influence represented by the attribute values of nodes.

A skyline community is defined as the largest k-core that is not dominated by other connected k-cores on multiple attributes in multi-valued networks. Intuitively, given two *k-core* subgraphs, H_1 and H_2; H_1 dominates H_2 if H_1 is not less than H_2 in all dimensions and is greater in at least one dimension, donated as $H_1 \prec H_2$. The value of H_1 or H_2 in each dimension is represented by the minimum value of the nodes in the *k-core*, with larger values being preferable in this paper. As shown in Figure 1(a), the multi-valued graph G contains 10 nodes, each of which has three values of different dimensions. Given $k = 2$, the subgraphs $H_1 = \{v_1, v_2, v_3\}$ with values (3, 7, 13) and $H_2 = \{v_3, v_4, v_5\}$ with values (2, 7, 8) are two 2-cores of G. According to the values, H_1 is better than H_2 in every dimension, we called that H_1 dominates H_2, denoted by $H_1 \prec H_2$. This example demonstrates how the skyline community is determined based on the dominance relationships between k-cores in the multi-valued networks.

Existing Solutions and Limitations. Existing approaches to the skyline community search problem [5][20][42] typically employ a candidate generation-and-verification computation paradigm. They first identify the candidate communities based on topological structure (i.e., *k-core*), and then perform dominance filtering on candidate communities considering attribute similarity. During the process, the space-partition techniques are developed to accelerate the computation. However, there are two limitations in existing approaches: 1) Limitation I, relating to the prevalence of redundant computations in the dominance filtering and search algorithms. Since the computation time complexity is exponential to the number of node attributes d, the approach is

* Jianye Yang is the corresponding author.



(a) Multi-valued Graph G.

(b) Skyline communities in G.

Figure 1: Example of multi-valued graph and skyline community.

rather inefficient for datasets with relatively large d. For example, it takes the state-of-the-art algorithm $10^4 \sim 10^5$ seconds to handle the Delicious dataset with independent attributes for $d = 5$. 2) Limitation II, the algorithm involves a significant amount of redundant calculations. The number of recursive calculations is largely related to the number of skyline community results, and in each recursive step, the algorithm performs a large number of repeated calculations, primarily equal to the number of nodes in the graph. For example, there are nearly 10^5 skyline community results in the Delicious dataset with independent attributes for $d = 5$, this requires approximately $10^5 \times |V|$ calculations.

Our Approach and Contributions. In this paper, we resort to deep learning techniques to address limitations on the model flexibility and computation efficiency of existing solutions. Specifically, we propose a novel framework, **Deep Skyline Community Search** (**DeepSCS**) to find the skyline communities accurately and efficiently. The overall architecture of DeepSCS is shown in Figure 2, which consists of two components, namely an offline pre-training phase and an online search phase. The offline pre-training phase pre-trains the skyline learnable graph convolutional network (SLGCN) that is designed specifically for SCS. Based on the pre-trained model, we search the skyline community for the given query node in online search phase.

Specifically, the SLGCN is designed to capture both topological and attribute-based skyline dominance characteristics of potential skyline communities. To facilitate training without labeled data and effectively capture dominance characteristics, we introduce two self-supervised loss functions specifically tailored for SCS, namely dominance loss and link loss. The dominance loss is motivated by the skyline property, which enables the model to learn representations that capture the dominance relationships between nodes. The link loss, inspired by the cohesiveness property of communities, guides the model to learn representations that capture the structural connectivity in the potential skyline communities. During the online search phase, we calculate the skyline community score by measuring the similarity between the representation of the query node and the representations of each node in the graph where the representations are inferred from the pre-trained SLGCN.

To address limitation II, we introduce an expected score gain function based online community search algorithm. Specifically, the community score is derived from the pre-trained SLGCN and reflects the likelihood of a node being included in the community. Based on this, we define an expected score gain function that quantifies the potential improvement in community quality with the addition of each node. Higher expected score gain values indicate more promising candidates for community membership. By scanning all nodes in the graph, we finish the skyline community retrieval for the query nodes.

Table 1: Symbols and Descriptions

Notation	Description
G	An undirected graph.
$V(G)$	A set of nodes in G.
$E(G)$	A set of edges in G.
$X \in \mathbb{R}^{\lvert V\rvert \times d}$	The feature matrix
$A \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert}$	The adjacency matrix
d	The number of node attributes
$H' \prec H$	H' dominates H
$q = V_q$	The query with node set V_q
$C_q, \widetilde{C}_q$	Ground-truth/predicted skyline community of q
$S \in \mathbb{R}^{\lvert V\rvert}$	The skyline community score vector
$f^{\theta}(\cdot)$	Skyline learnable graph convolutional network with parameters θ

Our extensive experiment study on 4 real-world datasets shows that our method can efficiently find skyline communities with high accuracy. Compared to the state-of-the-art skyline community search algorithm, our method can be up to 3 orders of magnitude faster when the number of node attributes reaches 5.

Our principle contributions are summarized as follows.

- We present a novel unsupervised skyline community search approach called DeepSCS, which consists of an offline pre-training phase and an online search phase.
- In the offline pre-training phase, we design an effective **S**kyline **L**earnable **G**raph **C**onvolutions **N**etwork (**SLGCN**).
- In the online search phase, we propose an efficient and effective search algorithm to find promising skyline communities.
- Through extensive experiments, we evaluate the efficiency and effectiveness of our DeepSCS approach for query dependence skyline community search over real-world datasets.

Roadmap. The rest of this paper is organized as follows. In Section 2, we introduce the problem definition and revisit the state-of-the-art algorithm. In Section 3, we present the overview of our approach DeepSCS, followed by the offline pre-training phase and online search phase in Section 4 and Section 5, respectively. In Section 6, we conduct extensive experiments. Section 7 and Section 8 review the related work and conclude this paper, respectively.

2 PRELIMINARIES

In this section, we introduce the problem definition and state-of-the-art algorithm. Table 1 summarizes the mathematical notations frequently used in this paper.

2.1 Problem Definition

In this paper, we consider a multi-valued graph $G = (V, E, X)$, where V and $E \subseteq V \times V$ denote a set of nodes and edges, respectively. $X \in \mathbb{R}^{|V| \times d}$ is the feature matrix to denote the d numerical attributes for nodes in V. In specific, for each node $v \in V$, we use a vector $X_v = (x_v^1, ..., x_v^d)$ to denote its d-dimensional features. Let $H = (V_H, E_H)$ be an induced subgraph of G, we define the value of H on the i-th dimension (for $i = 1, 2, ..., d$) as

$$f_i(H) = \min_{v \in V_H} \{x_i^v\}. \quad (1)$$

$A \in \mathbb{R}^{|V| \times |V|}$ is the adjacency matrix where the binary value $A_{ij} = 1$ indicates that there is an edge between v_i and v_j. The skyline community score vector is denoted by $S \in \mathbb{R}^{|V|}$. We use q and V_q interchangeably to indicate the query, while C_q and $\widetilde{C_q}$ represent the ground-truth and predicted community, respectively. Note that, a query may contain multiple nodes, and we aim to find a community covering all nodes in the query.

*Definition 2.1 (**Dominance relationship**). Let $H = (V_H, E_H)$ and $H' = (V_{H'}, E_{H'})$ be two communities. we call that H' dominates H, denoted by $H' \prec H$, if $f_i(H) \leq f_i(H')$ for all $i = 1, ..., d$, and there exists at least one dimension i such that $f_i(H) < f_i(H')$.*

*Definition 2.2 (**k-core**). Given a graph $G = (V, E, X)$, a k-core S of G is subgraph of G with each node in S having at least k neighbors in S.*

*Definition 2.3 (**Skyline Community Search, SCS**).* Given a graph $G = (V, E, X)$, an integer k, and query q, the problem of skyline community search aims to identify a query-dependent connected subgraph (i.e., community) $C_q = (V_{C_q}, E_{C_q}, X_{C_q})$, where nodes in the found community are an induced subgraph of G that satisfies the following properties.

- *Cohesive property: C_q is a connected k-core;*
- *Skyline property: there does not exist an induced subgraph C'_q of G such that C'_q is a k-core and $C'_q \prec C_q$;*
- *Maximal Property: there does not exist an induced subgraph C'_q of G such that (1) C_q is a connected k-core (2) C'_q contains C_q, and (3) $f_i(C'_q) = f_i(C_q)$ for all $1, ..., d$.*

Example 2.4. Consider the graph G shown in Figure 1(a). Suppose for instance that $k = 2$. Then, by Definition 2.3, the subgraphs $H_1 = \{v_1, v_2, v_3\}$ is a skyline community with values $f(H_1) = (3, 7, 13)$, because there does not exist other 2-core subgraph that can dominate it, and it is also the maximal subgraph that satisfies the cohesive and skyline properties. Similarly, $H_2 = \{v_6, v_8, v_9\}$ is a skyline community with $f(H_2) = (5, 2, 13)$. The subgraph $H_3 = \{v_6, v_8, v_9, v_{10}\}$ are not a skyline community, as $f(H_3) = (5, 2, 11)$ is dominated by H_2. The skyline communities in G are $\{v_1, v_2, v_3\}, \{v_6, v_8, v_9\}, \{v_8, v_9, v_{10}\}$, as shown in Figure 1(b).

Theorem 2.5 (**Skyline Community Dimensional Inclusion Property**). *Given a multi-valued graph $G = (V, E, X)$ and an integer k, let $SkyComm_d(G)$ be all skyline communities with d-dimensional features. Then, the set of skyline communities with d-dimensional features covers all skyline communities with $(d-1)$-dimensional features, i.e., $SkyComm_d(G) \supseteq SkyComm_{d-1}(G)$.*

Proof. *We prove the theorem by contradiction. Suppose H is a skyline community in $d-1$ dimensions with values $f_i(H)$ that cannot be obtained in $SkyComm_d(G)$ for all $i = 1, ..., d-1$. Since H is not in the union of all skyline communities in $d-1$ dimension, there must exist a skyline community H' in $d-1$ dimension such that: $f_i(H') \geq f_i(H)$ for all $i = 1, ..., d-1$. Furthermore, because H is a skyline community in $d-1$ dimensions, H cannot be dominated by any other k-core subgraph in $d-1$ dimensions. Hence, there exists at least one dimension j where: $f_j(H') > f_j(H)$. This would mean that H' dominates H in $d-1$ dimensions, contradicting the assumption that H is a skyline community in $d-1$ dimensions. Hence, H must be in the union of the skyline communities in $d-1$ dimensions when considering all the values f_d. By contradiction, we show that if H is a skyline community in $d-1$ dimensions, it must be included in the d-dimensional skyline communities. Therefore, $SkyComm_d(G) \supseteq SkyComm_{d-1}(G)$* □

Example 2.6. Consider the graph shown in Fig 1(a), the skyline community for 2-dimensional attributes is $\{v_8, v_9, v_{10}\}$ with the constrain $\{x_1 \geq 10, x_2 \geq 9\}$, which is included in the skyline community for 3-dimensional attributes. The skyline communities for 3-dimensional attributes in G is $\{v_1, v_2, v_3\}, \{v_6, v_8, v_9\}, \{v_8, v_9, v_{10}\}$, as shown in Fig 1(b) with the corresponding restrictions $\{x_1 \geq 3, x_2 \geq 7, x_3 \geq 13\}$, $\{x_1 \geq 5, x_2 \geq 2, x_3 \geq 13\}$, and $\{x_1 \geq 10, x_2 \geq 9, x_3 \geq 11\}$. Therefore, the constraint for the 2-dimensional skyline community can be computed simply by ignoring the x_3 constraint, and employing traditional skyline algorithms for dominance relationships.

According to Theorem 2.5, we prove that all skyline communities in high-dimensional space $(d-1)$ are contained by these in low-dimensional space (d). However, we observe that there also exists a phenomenon called *potential information loss*. In specific, if a query vertex q belongs to a skyline community in d dimensions but not in $d-1$ dimensions, it would result in an empty query outcome. This phenomenon is rather significant when the dimension gap exceeds 3. Despite this limitation, Theorem 2.5 facilitates the development of a single pre-trained model that encompasses all dimensions, allowing it to address any combination of dimensions within multi-valued graphs. Additionally, by training a node embedding on the d-dimensional attributes dataset, we can efficiently capture skyline communities across all dimensional combinations within d dimensions. This significantly simplifies the process of identifying communities across multiple feature dimensions since we do not need to develop separate models for each lower dimension.

2.2 Learnable Graph Convolutional Networks

Graph Convolutional Networks (GCN), as introduced by Kipf and Welling [17], have been proposed as a method to learn high dimensional node representations [37] by simultaneously capturing content features and structural topology. The fundamental principle underlying GCN is the utilization of localized aggregation of neighboring node features to iteratively update node representations. This approach enables the model to encode both the local and global graph structure, as well as node-specific attributes, into the node representation. However, the application of convolutional operations in generic graphs is hindered by the variable and unordered nature of neighboring units. To address these challenges, Gao et al. proposed Learnable Graph Convolutional Networks (LGCN) [11]. The core innovation of LGCN is to automatically select a fixed number of neighboring nodes for each feature based on value-based ranking in order to transform graph data into grid-like structures in 1-D format, thereby enabling the use of regular convolutional operations on generic graphs. By leveraging the graph's inherent structure, LGCN can effectively process and analyze complex relational data, making

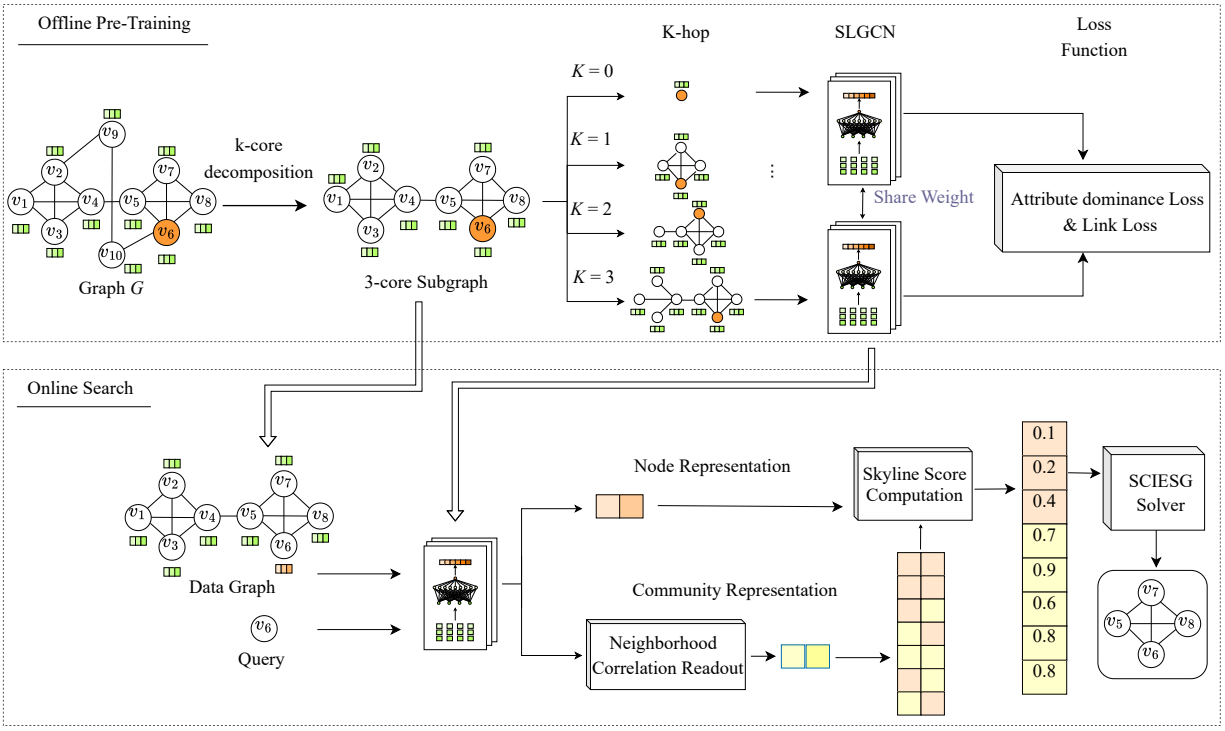


Figure 2: The architecture of proposed model.

them particularly well-suited for tasks involving multi-valued graphs. This offers a more flexible and adaptable approach to graph-based learning, potentially improving performance in scenarios where traditional GCN may be limited by the irregular structure of the input graph.

2.3 State-of-the-art

The state-of-the-art skyline community search algorithm employs recursive calculations to obtain the result, based on the dimension size of the feature, for $d = 2$ and $d \geq 3$.

For $d = 2$. The SkylineComm2D algorithm begins by computing the maximal *k-core* w.r.t. the second dimension x_2, which iteratively removes nodes with the smallest x_2 value until no *k-core* remains, ensuring that the resulting *k-core* possesses the maximum f_2 value among all *k-core*. Following this, the algorithm shifts its focus to finding the corresponding maximal f_1 value while maintaining the same f_2 value. This process is repeated until no valid *k-core* remains, and then a set of skyline communities is obtained. It is efficient, with a time complexity of $O(s(|V| + |E|))$ and space complexity $O(|V| + |E| + s)$, where s is the number of 2D skyline communities. However, the algorithm is inefficient when $d \geq 3$.

For $d \geq 3$. The Space-Partition algorithm reduces the dimension recursively and then calls the SkylineComm2D algorithm. This approach involves deriving all possible f_d values for the skyline communities and using these values to recursively partition the search space into regular subspaces. As a result, this method bears a time complexity exponential to d. Moreover, the Space-Partition algorithm is ineffective when the answer size is large.

Discussion. The existing skyline community algorithm bears major limitations on computation efficiency when the number of node attributes is relatively large (e.g., $d \geq 5$), or more tightly connected structural cohesiveness constraints exist in the graph. To address these limitations, we propose a deep learning based skyline community search, which can effectively capture complex numerical features and relationships within the graph. By leveraging the power of learnable graph convolutional network, our model aims to improve the efficiency and accuracy of skyline community search, even in high-dimensional spaces and large skyline community results.

3 OUR APPROACH DEEPSCS

Motivation. Because the traditional search algorithm for SCS is computationally expensive, in this paper, we recast the skyline community search as an unsupervised classification problem, which shifts the paradigm from traditional search algorithms to a machine learning-based approach by leveraging the power of unsupervised techniques to identify skyline communities in the multi-valued graph. More specifically, given a multi-valued graph $G(V, E, X)$, the objective of pre-training for SCS is to learn a generic encoder that can encode the skyline community dominance relationship and the graph topology into latent space. Then, for a given query, we can quickly identify the skyline community via the learned encoder.

Overview of DeepSCS. The overall architecture of DeepSCS is shown in Figure 2, which consists of two phases, namely offline pre-training and online search. In the offline phase, we pre-train a skyline learnable graph convolutions network (shortened as SLGCN), which is specifically designed to capture both the topological and attribute-based skyline dominance relationships of potential skyline communities. To this end, we propose a novel node dominance embedding that effectively identifies skyline community properties as defined in Definition 2.3. The loss function is a composite of two key components, namely dominance loss and link loss. In the online phase, we reformulate the skyline community search problem as a skyline community identification problem with expected score gain (shortened as SCIESG). This approach aims to identify communities that maximize the

skyline community score, which is derived from the pre-trained SLGCN representations. The SCIESG formulation seeks to find a connected subgraph that contains the query node, satisfies skyline community properties, and maximizes the expected score gain. The details of offline pre-training and online search are introduced in Section 4 and Section 5, respectively.

4 OFFLINE PRE-TRAINING

4.1 Overview

DeepSCS employs the learnable graph convolutional networks to learn the vertex embedding. This approach can preserve the subgraph relationships in the embedding space and support efficient, accurate skyline community candidate vertex retrieval. Specifically, given a multi-valued graph, we first generate the community-level subgraphs with the most consistent features from different views of a vertex in the graph as positive samples. Then, we utilize these samples for contrastive training. Next, we feed the augmented subgraph into a graph encoder (SLGCN) to extract latent features that encode community information and graph topology. The graph encoder generates both vertex-level and community-level representations. Finally, we use these learned representations for the loss computation, which includes dominance loss and link loss. The resulting loss is back-propagated to update the parameters in the SLGCN. This process enables our model to iteratively enhance the capacity to capture and represent the complex relationships within the graph structure.

4.2 Augmented Subgraph Construction

Motivation. The unsupervised learning is based on the augmented subgraphs. To capture the rich graph structure information conforming to the *cohesive property* and *maximal property*, we propose a two-step subgraph construction method as shown in Figure 2. We first compute the k-core subgraphs, then extract the K-hop neighborhood subgraphs within the k-core subgraphs.

The two-step subgraph construction strategy is based on the following facts. On the one hand, real-world graphs are usually large in scale. Thus, it is important to reduce the size of the graph by pruning the unrelated vertices in advance. On the other hand, not all vertices in the graph are contained in the skyline communities based on the skyline community property. Therefore, to reduce the training cost, we start by conducting k-core decomposition on the original graph. After that, we extract the K-hop neighborhood subgraphs in the k-core subgraphs. Each K-hop corresponds to a different neighborhood structure, allowing the model to aggregate information from neighbors from different distances. This is important because relationships and dependencies in a graph often exist at multiple scales. A vertex might provide insights into its attribute dominance, which may affect not only its immediate neighborhood but also the distant ones. To better understand how each vertex impacts the others, the K-hop approach helps capture this information. To strike a balance between the search space and personalization, we set the upper limit for K at 3, as suggested by the experimental results.

4.3 SLGCN Architecture

Motivation. The graph encoder generates both vertex-level and community-level representations with the input augmented subgraphs. GCN is a powerful tool for processing graph-based data, offering an effective way to learn the representations of nodes in a graph while considering the graph's structure. In this paper, we follow the state-of-the-art learnable graph convolutional network (LGCN) [11], which transforms generic graphs into data using a novel k-largest node selection process that utilizes ranking among node feature values. This innovative technique allows for modifying the node selection and ranking processes to prioritize nodes that are likely to form skyline communities.

Algorithm 1: Forward Propagation

Input: center node v, feature matrix X, adjacent matrix A, LGCN layers L.
Output: The node representation Z_v^{node} and community-level representation Z_v^{comm}

$\mathcal{X}_v \leftarrow \{x_v^0, x_v^1, ..., x_v^K\}$
$h_v^{(0)} \leftarrow \mathcal{X}_v W$
for $l = 0, ..., L-1$ **do**
  $\hat{h}_v^{(l)} \leftarrow g(h_v^{(l)}, A, k)$
  $h_v^{(l+1)} \leftarrow c(\hat{h}_v^{(l)})$
$Z_v^{node} \leftarrow {}^0h_v^{(L)}$; $Z_v^{comm} \leftarrow \{\}$
for $k = 1, ..., K$ **do**
  $agg_v^k \leftarrow \sum_{u \in N(v)} \frac{{}^kh_v^{(L)^T} \cdot h_u^{(L)}}{\|{}^kh_v^{(L)}\| \|{}^kh_u^{(L)}\|}$
  $Z_v^{comm} \leftarrow Z_v^{comm} + agg_v^k \cdot {}^kh_v^{(L)}$
return Z_v^{node}, Z_v^{comm}

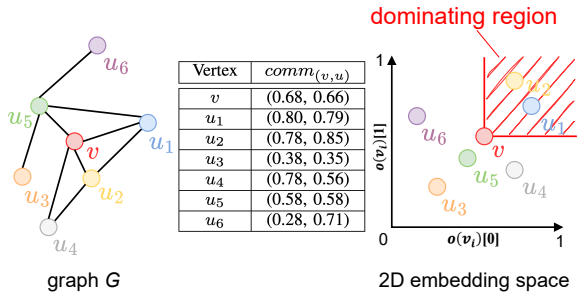
Technical Details. Based on the state-of-the-art LGCN [11], in this paper, we propose SLGCN as the graph encoder in DeepSCS. In general, the graph encoder takes a k-core subgraph as input and produces both node-level and community-level representations. Algorithm 1 summarizes the forward propagation process. Specifically, we implement a propagation procedure to obtain the token sequence of the K-hop neighborhood matrices, which is represented as $\mathcal{X}_K = \hat{A}^K\mathcal{X}$. Here, $\hat{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ denotes the normalized adjacency matrix, $\hat{A}^K$ represents the K-hop neighborhood adjacency matrix, and D is the degree matrix of A. We then assemble the aggregated neighborhood sequence of node v from $\mathcal{X}_v = \{x_v^0, x_v^1, ..., x_v^K\}$, where $\mathcal{X}_v \in \mathbb{R}^{(K+1)\times d}$.

In the following, we project $\mathcal{X}_v$ to the hidden dimension d_m of the GCN using a learnable linear projection. This is expressed as $h_v^{(0)} = \mathcal{X}_v W$ where $W \in \mathbb{R}^{d \times d_m^{(0)}}$, and $h_v^{(0)} \in \mathbb{R}^{(K+1)\times d_m^{(0)}}$. Subsequently, we pass $h_v^{(0)}$ through L LGCN encoder layers. To enhance the stability and performance of our model, we apply layer normalization (LN) before each block of the LGCN encoder.

LGCN encoder layer. An LGCN encoder layer consists of two components: k-largest node selection denoted as $g(\cdot)$ and GCN denoted as $c(\cdot)$. In layer l, $h_v^{(l)}$ is initially processed by the $g(\cdot)$ and then fed into $c(\cdot)$. The k-largest node selection transforms the graph data into a grid-like structure by selecting the k nodes with the highest feature values for each feature dimension. A graph convolutional neural network is applied to aggregate information from the selected nodes and produce new node representations as follows.

$$\begin{aligned} \hat{h}_v^{(l)} &= g(h_v^{(l)}, A, k) \\ h_v^{(l+1)} &= c(\hat{h}_v^{(l)}) \end{aligned} \tag{2}$$

Finally, we obtain the node representation $Z_v^{node} = {}^0h_v^{(L)} \in \mathbb{R}^{d_m^{(L)}}$ with the latent representation $h_v^{(L)} \in \mathbb{R}^{(K+1)\times d_m^{(L)}}$ obtained



Vertex	$comm_{(v,u)}$
v	(0.68, 0.66)
u_1	(0.80, 0.79)
u_2	(0.78, 0.85)
u_3	(0.38, 0.35)
u_4	(0.78, 0.56)
u_5	(0.58, 0.58)
u_6	(0.28, 0.71)

Figure 3: Example of node dominance embedding.

after L layers. Additionally, we have the latent representations of the neighborhood tokens $\{{}^1h_v^{(L)}, ..., {}^Kh_v^{(L)}\}$.

Aggregation with Neighborhood Correlation readout. We consider the correlation between the neighborhood of each node and the representation of the node itself using the cosine similarity [36]. We also take the weighted neighborhood tokens to obtain the representation at the community level.

$$agg_v^k = \sum_{u \in N(v)} \frac{{}^kh_v^{(L)T} \cdot h_u^{(L)}}{\|{}^kh_v^{(L)}\| \|{}^kh_u^{(L)}\|}$$
$$Z_v^{comm} = \sum_{k=1}^{K} agg_v^k \cdot {}^kh_v^{(L)} \tag{3}$$

Here, $||$ indicates the norms of the feature vectors.

4.4 Training Target

Motivation. In the context of the skyline community, nodes are assessed based on both graph cohesion and attribute dominance within their communities to capture node dominance representation. Our SLGCN method aims to capture the dominance relationship between a node v and its neighbor u in the embedding space. This embedding strategy is designed to retain the *skyline property* and *maximal property* within the skyline community. Furthermore, we also introduce a link loss function to enable the *cohesive property* within the skyline community. This function encourages connected nodes to have similar latent representations, while simultaneously pushing apart the representations of unconnected nodes. The link loss contributes to the preservation of local graph topology, while the dominance loss captures global information and dominance relationships. By combining these loss functions, we aim to create a comprehensive embedding that encapsulates both local and global graph properties. Judiciously, this dual approach allows us to effectively capture the properties of cohesiveness, skyline, and maximality that are defined on the skyline community.

Loss Function. We design the dominance loss by the margin triplet loss [29], which ensures that the dominance relationship between a selected node with its corresponding community and the other node from the communities. The representation of dominance relationship between two nodes v and u in a community is defined as below.

$$comm_{(v,u)} = Z_v^{\text{node}} Z_u^{\text{comm}} \tag{4}$$

Then, following the approach in [41], we calculate the dominance loss as follows.

$$\mathcal{L}_d = \sum_{v,u \in E} \max(0, \sigma(comm_{(v,v)}) - \sigma(comm_{(v,u)}) + \epsilon) \tag{5}$$

Algorithm 2: Offline Pre-training

Input: The data graph G, batch size n_{batch}, layer number L, SLGCN $f^\theta(\cdot)$, learning rate η, coefficient α
Output: a pre-trained SLGCN $f^\theta(\cdot)$

```
1  Initialize optimizer opt_θ with learning rate φ;
2  Separate V into batches {V_b} with the batch size n_batch;
3  for each v ∈ V do
4      G_v ← augmented subgraph sampler;
5  for {V_b} ∈ V do
6      for each v ∈ V_b do
7          comm_(v,v) ← f^θ(v, X(G_v), A(G_v), L)
8      for each u, v ∈ V_b do
9          L_d = max(0, σ(comm_(v,v)) − σ(comm_(v,u)) + ε)
10         L_k = −A(u,v)(comm_(v,v)) + (1 − A(u,v))((comm_(u,v)))
11         L += L_d + αL_k
12     Update θ by opt_θ with loss L / |V_b|^2
13 return pre-trained SLGCN f^θ(·);
```

where ϵ and $\sigma(\cdot)$ are the margin value and the sigmoid function, respectively.

Intuitively, when the loss function $\mathcal{L}_d = 0$, the embedding vector $\sigma(comm_{(v,v)})$ is *dominating* (or equal to) $\sigma(comm_{(v,u)})$. Thus, we simply say that $\sigma(comm_{(v,v)})$ dominates $\sigma(comm_{(v,u)})$ (denoted as $\sigma(comm_{(v,v)}) \prec \sigma(comm_{(v,u)})$). Moreover, SLGCN guarantees that the embedding follows the dominance relationship (i.e., $\sigma(comm_{(v,v)}) \prec \sigma(comm_{(v,u)})$).

Example 4.1. Figure 3 illustrates an example of the node dominance embedding between node v and its neighbor nodes u. Each node has a 2D embedding vector $comm_{(v,v)}$ via SLGCN. For example, as shown in tables, we have $comm_{(v,v)} = (0.68, 0.66)$ and $comm_{(v,u_1)} = (0.80, 0.79)$. We plot the embedding vectors of nodes in a 2D embedding space on the right side of the figure. We can see that $comm_{(v,v)}$ is dominating $comm_{(v,u_1)}$ and $comm_{(v,u_2)}$, which means that v is potentially a skyline constraint of $comm_{(v,u_1)}$ and $comm_{(v,u_2)}$. On the other hand, since $comm_{(v,v)}$ is not dominating $comm_{(v,u_3)}$ in the 2D embedding space, node v cannot be the skyline constraint for $comm_{(v,u_3)}$ in the graph G.

Furthermore, we apply a link loss to improve node connectivity. The link loss ensures neighboring nodes are similar while distinguishing non-adjacent nodes, as formulated below.

$$\mathcal{L}_k = \frac{1}{|V|^2} \sum_{v,u \in E} (-A(u,v)(comm_{(v,v)}) + (1 - A(u,v))(comm_{(u,v)})) \tag{6}$$

The overall loss function of SLGCN is defined as:

$$\mathcal{L} = \mathcal{L}_d + \alpha \mathcal{L}_k \tag{7}$$

where $\alpha \in [0, 1]$ is the coefficient to balance two losses.

Algorithm 2 illustrates the training procedure, which incorporates the loss function defined in Equation 7. We begin by initializing the optimizer parameters and by partitioning all nodes into several batches (Lines 1-2). This batching approach allows for efficient processing of large graphs. Subsequently, we sample all augmented subgraphs (Lines 3-4). This sampling step is crucial for capturing diverse local structures within the graph. We then train *SLGCN* batch by batch to obtain both node-level

and community-level representations for all nodes (Lines 6-7). This dual-level representation enables our model to capture both fine-grained node features and broader community structures simultaneously. With these representations, we compute the composite loss, which encompasses both dominance and link losses (Lines 8-11). We update the parameters in *SLGCN* via the computed loss (Line 12). This update step allows the model to iteratively refine its ability to capture and represent the complex relationships within the graph structure. Finally, we return a pre-trained SLGCN $f^{\theta}(\cdot)$ (Line 13). By implementing this training procedure, we aim to optimize our model's performance in capturing both local and global graph properties, essentially for effective skyline community detection.

5 ONLINE SEARCH

5.1 General Idea

The overall computation paradigm of online search phase is illustrated in the bottom part of Figure 2. In specific, with the learned model, we first compute the skyline community score for each vertex in the graph by calculating the pairwise similarity with the query nodes. Then, we use an expected score gain based method to collect the resulting vertices which are considered as the output community. Below is a concrete example for the online search process.

Example 5.1. Consider the data graph and query illustrated in Figure 2. With the learned model, we compute the community score of the 8 *vertices, i.e.,* $S = (0.1, 0.2, 0.4, 0.7, 0.9, 0.6, 0.8, 0.8)$. *Next, we begin by adding the query node* v_6 *to the identified community* $\widetilde{C}_q$. *At this point,* $\widetilde{C}_q$ *contains only the query node, and the ESG (Expected Skyline Gain) is calculated as* $\frac{1}{1^{0.5}}(0.6 - \frac{4.5}{8} \times 1) = 0.0375$. *After that, we iteratively consider the neighboring vertices. For a neighboring vertex* v, *we add it to the result community if it can increase the ESG score. By continuing this process, we can obtain a resulting skyline community consisting of nodes* v_5, v_6, v_7, *and* v_8.

5.2 Skyline Community Score Computation

Motivation. *SLGCN* model is designed to capture both the skyline community dominance relationship and graph topology in the latent representation. However, it is also important to make sure that the latent representation can reflect the underlying graph structure effectively. Intuitively, similar nodes in the graph should have similar node representations, maintain proximity in the graph structure, and share comparable community-level information. Leveraging this principle, we compute the skyline community score by evaluating the similarity for the representations of the query node and other nodes in the graph. In our approach, nodes that are likely to be part of the resulting skyline community should demonstrate higher similarity. By employing this similarity-based scoring mechanism, we are able to effectively retrieve skyline community members by quantifying their similarity to the query node in the learned embedding space. This approach aligns with our goal of developing a robust and effective method for identifying meaningful community structures in multi-valued graphs.

Algorithm Details. Algorithm 3 summarizes the details of our skyline community score computation method. It takes three parameters as input, including the query nodes, the graph, and the pre-trained SLGCN. It generates the skyline community score with respect to the given query as output. Initially, the skyline community score is set to zero for all nodes in G (Line 1). Then, for each node v, we calculate the average similarity between v and all query nodes (Lines 3-5). Here, we employ cosine similarity as our primary metric. The resulting scores are normalized to the range $[0, 1]$ to ensure consistent comparison across queries of different sizes (Line 5).

Algorithm 3: Skyline Community Score Computation

Input: The query V_q, graph G, and SLGCN $f^{\theta}(\cdot)$
Output: The skyline community score S

1 Initialize $S \leftarrow \{s_v = 0 \text{ for } v \in V\}$
2 **for** $v \in V$ **do**
3 **for** $u \in V_q$ **do**
4 $s_v \leftarrow s_v + \frac{\sum_{i=0}^{d_m^{(L)}} f_i^{\theta}(v) f_i^{\theta}(u)}{\sqrt{f_i^{\theta}(v) f_i^{\theta}(v)} \sqrt{f_i^{\theta}(u) f_i^{\theta}(u)}}$
5 $s_v \leftarrow \frac{s_v}{|V_q|}$
6 **return** S;

5.3 Skyline Community Search

Motivation. The community score quantifies the likelihood of a node being included in the community. An ideal community is one where all nodes exhibit high community scores with the query nodes. Existing unsupervised learning-based community search methods typically utilize a threshold-based strategy. That is, nodes having a community score larger than the threshold are included in the resulting communities. However, the approach of using a fixed threshold or a fixed number of nodes would potentially lead to inaccurate community structure. To address this limitation, we introduce the expected score gain (ESG) function, which is defined as the sum of nodes scores in the community subtracted by the sum of expected scores under random node selection. In this framework, communities that maximize the ESG are considered high-scoring. This concept is inspired by the well-established metric of community cohesiveness known as density modularity, as proposed in the context of density modularity based community search [16]. Density modularity measures the number of edges in the community minus the expected number of edges in the community if the edges were randomly distributed. A higher density modularity indicates a more cohesive community. Compared to the fixed threshold approach, this approach provides a more flexible way for community identification.

Technical Details. Next, we begin with the metric definition of the expected score gain, and then introduce the details community search algorithm based on this metric.

*Definition 5.2 (**Expected Score Gain, ESG**). Given a graph* $G(V, E, X)$, *a community* $C = (V_C, E_C)$ *and the community score* S, *the expected score gain of* C *is defined as:*

$$ESG(S, C, G) = \frac{1}{|V_C|^r}\Big(\sum_{v \in V_C} s_v - \frac{\sum_{u \in V} s_u}{|V|}|V_C|\Big) \tag{8}$$

where $r \in [0, 1]$ *is a hyer-parameter to control the granularity of the subgraph, and a higher* r *leads to a more fine-grained one.*

In Equation 8, the first term $\sum_{v \in V_C} s_v$ is the sum of the skyline community score of the nodes in the selected community. The second term $\frac{\sum_{u \in V} s_u}{|V|}|V_C|$ is the expected skyline community score by considering the average graph score.

With the concept of ESG, we now aim to identify communities that not only maximize the community score but also are

Algorithm 4: Skyline Community Search

Input: The community score S, multi-valued graph G, d and query V_q

Output: The identified community $\widetilde{C}_q$

```
1  C, C̃_q, Q ← V_q
2  maxESG ← −∞
3  while |Q| < |V| do
4      Q̄ = {v ∈ V \ Q | ∃N(v) ∩ Q}
5      u ← argmax_{v∈Q̄}
6      Q ← Q ∪ u
7      if ESG(S, C̃_q ∪ {u}, G) > maxESG then
8          for i in d do
9              if f_i(u) < f_i(C) then
10                 f_i(C) ← f_i(u)
11         maxESG ← ESG(S, C̃_q ∪ {u}, G)
12 C̃_q ← C̃_q ∪ {u ∈ G | f_i(u) >= f_i(C) for i in d}
13 return C̃_q
```

structurally cohesive fulfill the properties of a skyline community. Besides, the identified community is query-dependent and therefore should contain the query nodes.

Community Search. The overall algorithm of ESG-based *Skyline Community Search* is summarized in Algorithm 4, which takes the community score, multi-value graph and query nodes as input, and outputs the retrieved skyline community. In specific, we use a set C to maintain the nodes with the minimal constraints in $\widetilde{C}_q$, a set $\widetilde{C}_q$ to keep track of candidate nodes, and a set Q to store the traversed nodes, which are all initialized as V_q. Then, the maximum expected score gain is initialized to negative infinity. (Line 1-2). The algorithm proceeds until all nodes have been traversed or terminates prematurely when no promising candidates remain (Lines 3-11). In each iteration, we select the node $\overline{Q} = \{v \in V \setminus Q | \exists N(v) \cap Q\}$ with the highest community score that satisfies two criteria: it has not been traversed and it is located at the boundary of the traversed node set (Line 4-6). If the merging of selected nodes increases the expected score gain of the previous intermediate subgraph, then we update the constraint values across all dimensions, as well as the maximum expected score gain. Otherwise, the algorithm terminates prematurely (Lines 7-11). Finally, find all vertices that satisfy the constraints and the skyline community is returned (Lines 12-13).

5.4 Time Complexity

Time complexity of pre-training. The projection of three matrices incurs a time complexity of $O(3d_h^2(K+1))$, where d_h is the dimension of the hidden layer of LGCN and K is the number of hops in the neighborhood aggregation. The dot product between query and key takes $O(d_h(K+1)^2)$ time. Consequently, the overall time complexity of the LGCN layer is $O(3d_h^2(K+1)+d_h(K+1)^2)$. Since there are $|V|$ nodes in the graph and our architecture employs L LGCN encoder layers, the time complexity of *SLGCN* for a single forward pass is $O(L \times |V| \times (3d_h^2(K+1) + d_h(K+1)^2))$. Thus, the total time complexity of pre-training is $O(t \times L \times |V| \times (3d_h^2(K+1) + d_h(K+1)^2))$ if we train the model for t epochs.

Time complexity of community score computation. The time complexity for calculating a pair-wise similarity is $O(d_h)$. Therefore, the overall time complexity for computing the community score is $O(|V_q| \times |V| \times d_h)$ since we need to consider each pair of nodes in V_q and V.

Time complexity of Skyline Community Search. The time complexity for searching nodes with the maximum score is $O(d \times |V| \times \log|V|)$. This operation may be executed up to $|V|$ times in the worst case. Thus, the overall time complexity of skyline community search is $O(d \times |V|^2 \times \log|V|)$.

Table 2: Dataset

Network	$\|V\|$	$\|E\|$	d_{max}	k_{max}
Slashdot	70,068	358,647	2507	54
Delicious	536,108	1,365,961	3216	33
Lastfm	1,191,805	4,519,330	5150	70
Flixster	2,523,386	7,918,801	1474	68
GenCAT	16,384	133,703	255	36

Table 3: Hyper-parameter

Parameters	Values
Loss balancer α	0.01, **0.1**, 0.3, 0.5, 0.7, 0.9
Subgraph granularity τ	0.1, 0.3, **0.5**, 0.7, 0.9
Hop numbers	1, 2, **3**, 4, 5, 6
Epoch numbers	25, 50, **100**, 150, 200

5.5 Discussion

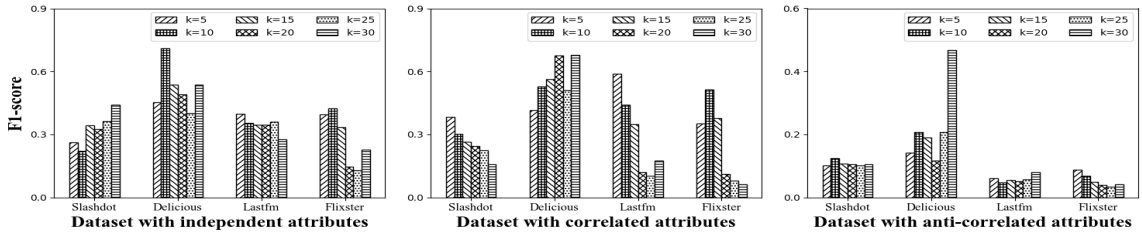
It is worth mentioning that our techniques can be easily extended to other graph models, such as property graphs or temporal graphs[38, 39]. The property graphs can be easily converted to the attributed graph since the properties can be simply considered as the node attributes. As for the temporal graphs, we can convert the timestamps or intervals into a relative attribute by applying techniques such as Fourier Transform to represent temporal patterns in a frequency domain.

6 EXPERIMENTS

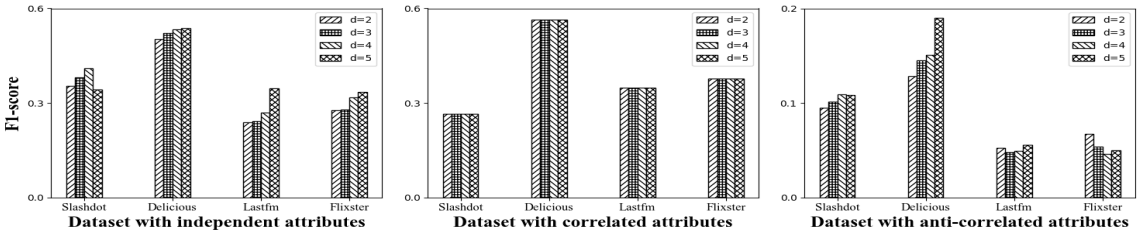
6.1 Experimental Setup

Dataset. We use four real-world datasets (i.e., Slashdot, Delicious, Lastfm, and Flixster) and a synthetic dataset GenCAT [23] to evaluate our proposals. The characteristic of the datasets are summarized in Table 2, where d_{max} and k_{max} denote the maximal degree and the maximal core number of the network, respectively. The real datasets are downloaded from http://networkrepository.com. It is worth mentioning that the original datasets do not contain numerical attributes. In order to evaluate the performance of our algorithms, we apply the widely used method in the skyline algorithm [7] to generate the numerical attributes for our datasets. Following the existing study [7], we generate three different types of numerical attributes, namely *independence, correlation,* and *anti-correlation. Independence* implies that the attribute values are generated independently using a uniform distribution. *Correlation* means that if a node performs well in one dimension, it also performs well in other dimensions. *Anti-correlation* indicates that if a node performs well in one dimension, then it is bad in one or all other dimensions. Intuitively, the number of skyline communities in a network with correlated attributes should be significantly smaller than the number in the same network with independent or anti-correlated attributes.

Baseline. We use the state-of-the-art traditional algorithm SC-CP [20] and the state-of-the-art learning based community search method TransZero[34] as the baselines for performance study. Note that SC-CP is an exact method.

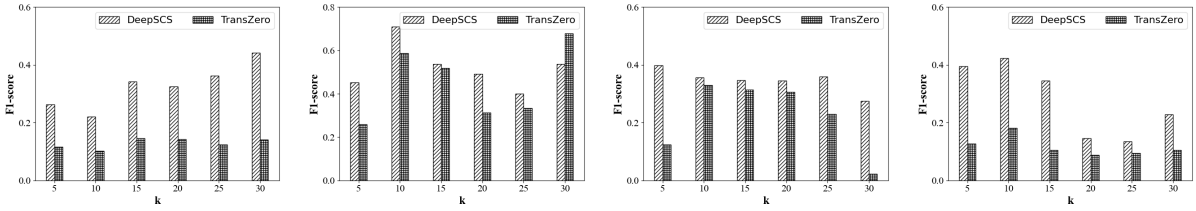


(a) F1-score results (Vary k, $d = 5$)



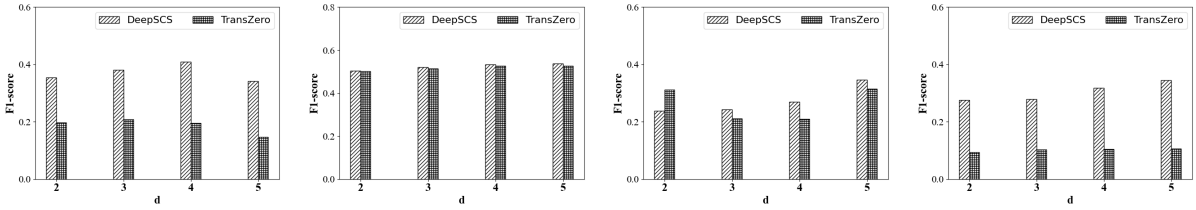
(b) F1-score results (Vary d, $k = 15$)

Figure 4: F1-score results under different settings



(a) Slashdot (b) Delicious (c) Lastfm (d) Flixster

Figure 5: F1-score of DeepSCS and TransZero in networks with independent attributes (vary k, $d = 5$)



(a) Slashdot (b) Delicious (c) Lastfm (d) Flixster

Figure 6: F1-score of DeepSCS and TransZero in networks with independent attributes (vary d, $k = 15$)

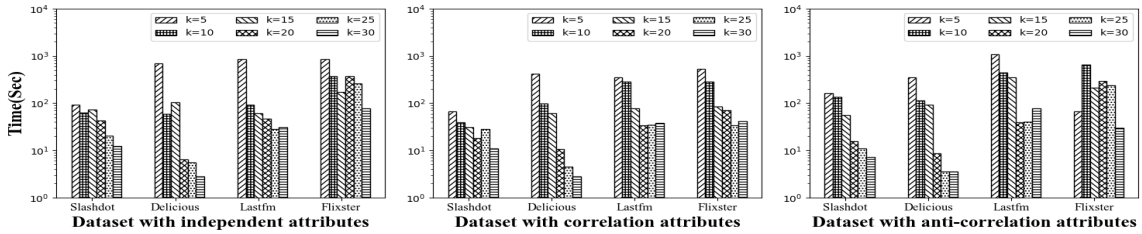
Query Generation. We generate all queries randomly from the available ground-truth communities which are combinations of all satisfied skyline communities. Our experimental design incorporates 100 training queries and 100 testing queries. Following the approach outlined in [34], we randomly select 1 to 3 nodes from the ground-truth communities to serve as query nodes.

Metrics. Following existing studies [15, 19], we use F1-score [28] to evaluate the accuracy of the algorithms. This metric provides a balanced measure of precision and recall. We also report the elapsed running time for training and search processing.

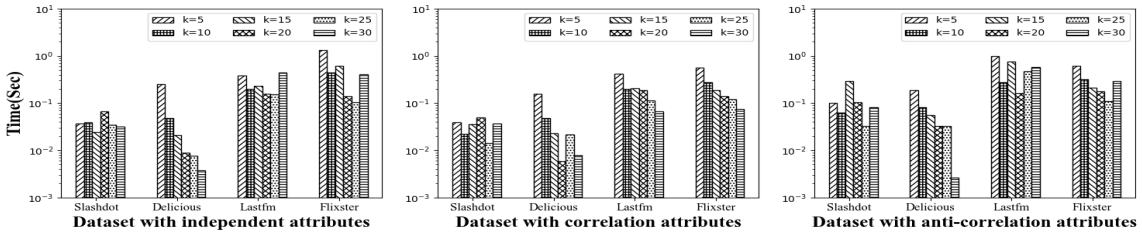
Implementation Details. The experimental setup for DeepSCS involves running the model for 100 epochs with an implemented early stopping mechanism. The batch size is dynamically set to the number of nodes in the graph, with an upper limit of 8000 to accommodate memory constraints. A dropout rate of 0.1 is employed to mitigate overfitting. The augmented subgraph sampler utilizes a maximum of 5 hops. The model architecture incorporates 8 attention heads, consistent with previous research. The temperature parameter τ is fixed at 0.5 in all datasets, while the balance coefficient α is set to 0.1. The number of LGCN layers is follows the configuration described in [11]. To ensure computational feasibility, the search space is limited to a maximum of 50% of the total number of nodes, not exceeding 10,000 nodes. All experiments are conducted on a high-performance server equipped with an Intel(R) Xeon(R) Gold 6342 CPU, 503GB of memory, and an Nvidia RTX 4090 GPU.

6.2 Effectiveness Evaluation

Exp-1: Effect of k on accuracy ($d = 5$). We evaluate the accuracy of DeepSCS in Figure 4(a) by varying the core number k from 5 to 30. The results reveal that the F1-score of DeepSCS demonstrates independence from variations in k-core. This phenomenon can be attributed to several factors. Firstly, the graph structure is modified according to the value of k, requiring the model to be retrained for each specific value of k. Secondly, the number of ground-truth communities exhibits a positive correlation with increasing k values, requiring the online search phase

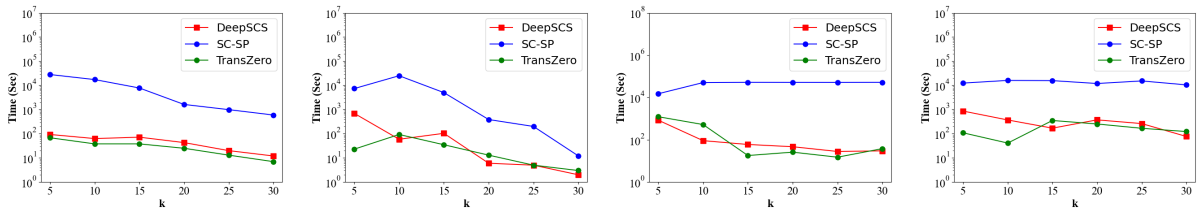


(a) Efficiency results of the training phase



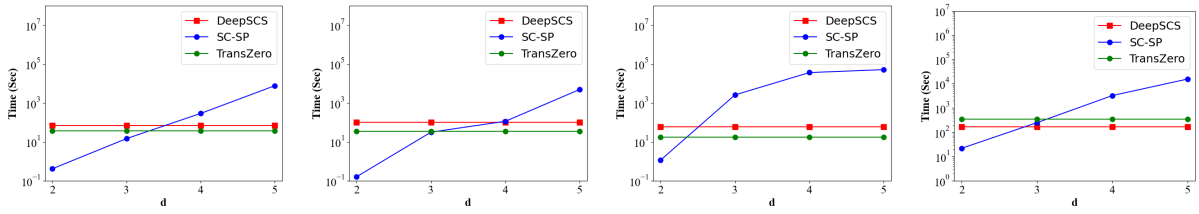
(b) Efficiency results of the search phase

Figure 7: Efficiency results



(a) Slashdot (b) Delicious (c) Lastfm (d) Flixster

Figure 8: Efficiency of DeepSCS, SC-CP, and TransZero in networks with independent attributes (vary k, $d = 5$)



(a) Slashdot (b) Delicious (c) Lastfm (d) Flixster

Figure 9: Efficiency of DeepSCS, SC-CP, and TransZero in networks with independent attributes (vary d, $k = 15$)

to adapt to predict the community size. Furthermore, mid-range k values (15-20) frequently yield optimal performance, with the F1-score of DeepSCS reaching over 70%.

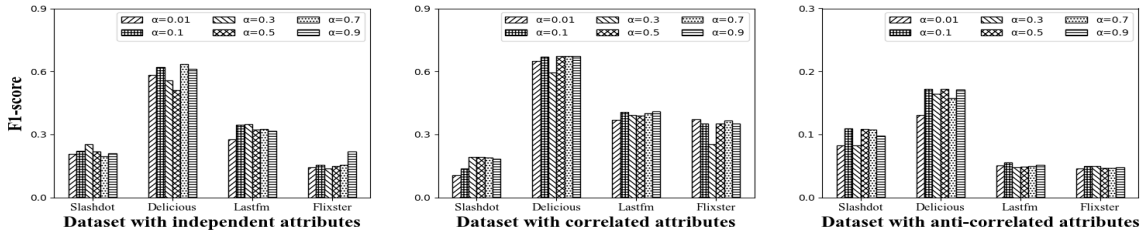
Exp-2: Effect of d on accuracy ($k = 15$). We evaluated the F1-score by varying the dimension d from 2 to 5. The results for all datasets are presented in Figure 4(b). We observed that the F1-score increases as d increases for both independent and anti-correlated attributes, which is consistent with Theorem 2.5. Furthermore, a significant finding is the consistency of the F1-score across all dimensions for correlated attributes. We attribute this phenomenon to the invariance of ground-truth skyline communities in the presence of correlated attributes. Basically, under the skyline community constraint, graphs with correlation attribute are less likely to have multiple skyline communities.

Exp-3: Accuracy comparison with baseline. We also compared the accuracy of DeepSCS and TransZero in Figure 5 and Figure 6. As reported, DeepSCS outperforms TransZero generally. This is because TransZero does not consider the skyline and dominance properties during its training phase. These findings further validate the effectiveness of DeepSCS.

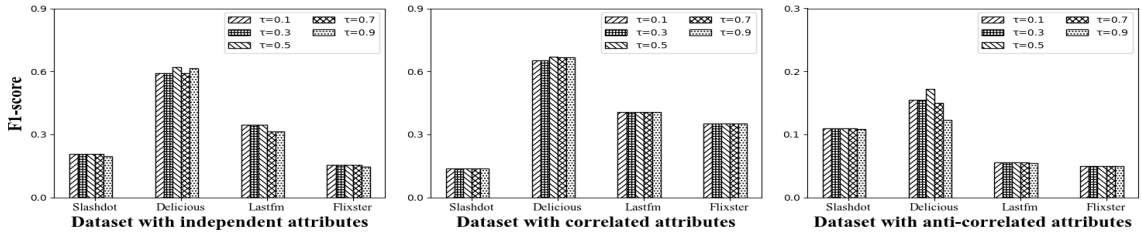
6.3 Efficiency Evaluation

Exp-4: Effect of k on efficiency ($d = 5$). In Figure 7, we report the efficiency results, including the efficiency of both the training and the search phase. Figure 7(a) shows the efficiency of the training phase, It is reported that the training time decreases with increasing k for all datasets and almost all types of attributes. This observed trend aligns with theoretical expectations, as higher k-cores represent more constrained subgraphs, indicating that less data is used for calculations. Figure 7(b) shows the efficiency of the search phase, for all types of attributes, the search time remains relatively stable across all k values, with only minor fluctuations in Slahdot, Lastfm, and Flixster. Furthermore, during both the training and testing phases, the time spent on the Delicious dataset decreases significantly as k increases.

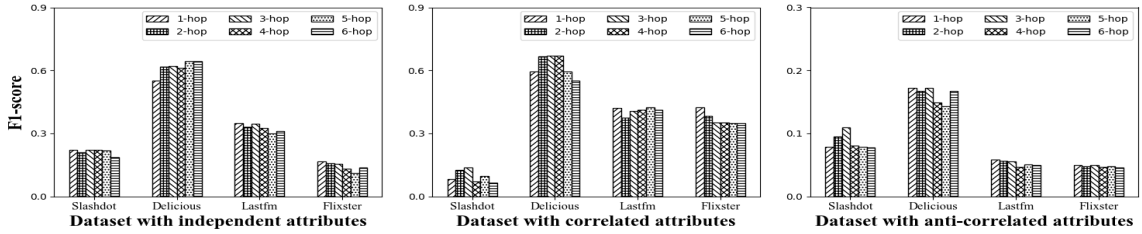
Exp-5: Efficiency evaluation of all algorithms (vary k, $d = 5$). For $d = 5$, the efficiency tests of DeepSCS, SC-CP, and TransZero



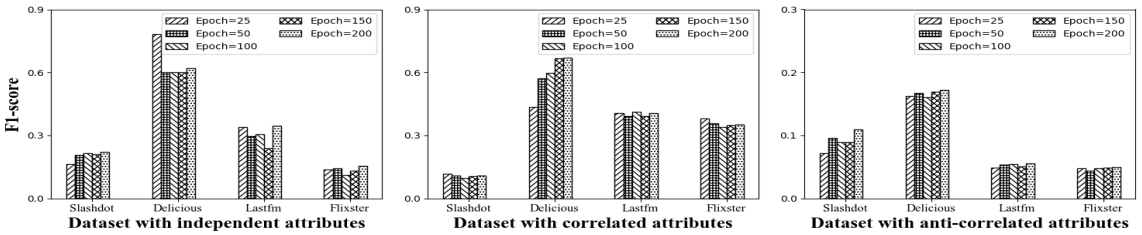
(a) F1-score with varying α



(b) F1-score with varying τ



(c) F1-score with varying hop numbers



(d) F1-score with varying epoch numbers

Figure 10: Hyper-parameter analysis results

in the networks with independent attributes are reported in Figure 8. As observed, the running time for DeepSCS, SC-CP, and TransZero gradually decreases as k increases. This is because, as k increases, the number of vertices within the k-core of the network gradually decreases. Thanks to the powerful GCN, DeepSCS and TransZero are observed much faster than SC-CP, achieving an average speedup of 10.6× and up to 1739.6× across all datasets compared to SC-CP.

Exp-6: Efficiency evaluation of all algorithms (vary d, $k = 15$). We vary d from 2 to 5 to evaluate the processing time of DeepSCS, SC-CP, and TransZero on the four networks with independent attributes and $k = 15$. The results in Figure 9 show that, for $d \geq 3$, although the performance of DeepSCS and TransZero is worse than SC-CP in small datasets, DeepSCS and TransZero still outperforms SC-CP in large datasets. The running time of SC-CP increases as d increases. In contrast, the running time of DeepSCS and TransZero remains constant across all datasets, regardless of the increase in d.

6.4 Hyper-parameter Analysis ($d = 5$, $k = 15$)

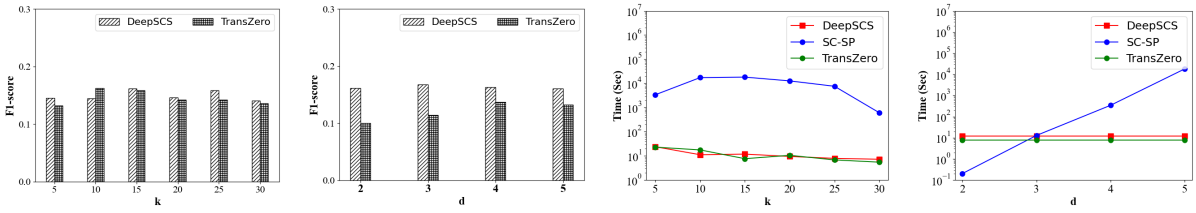
Exp-7: Varying α. Figure 11(a) evaluates the effect of α, which serves as a weighting factor that determines the relative contribution of the dominance loss and link loss to the overall optimization objective in Equation 7. Compared to the correlated ones, datasets with independent attributes exhibit high sensitivity to the variations of α. The method performs poorly on datasets with anti-correlated attributes. In general, $\alpha = 0.1$ can strike a good balance for the two losses.

Exp-8: Varying τ. In Figure 11(b), we evaluate effect of τ. As defined in Definition 5.2, τ serves to regulate the granularity of the subgraph structure. A larger τ results in more fine-grained subgraph representations. On Delicious, the performance of DeepSCS improves with the increase of τ across all types of attributes. On the other datasets, the performance decreases with the increase of τ. In general, $\tau = 0.5$ demonstrates good performance.

Exp-9: Varying hop numbers. Figure 11(c) reports the effect of the number of hops. As reported, DeepSCS performs relatively

Table 4: Ablation Study

Models	Slashdot	Delicious	Lastfm	Flixster	GenCAT	Average +/-
Full model	0.3423	0.5367	0.3469	0.3454	0.1604	-
w/o $\mathcal{L}_d$	0.1395	0.5035	0.3024	0.0679	0.1773	-29.61%
w/o $\mathcal{L}_k$	0.2368	0.5233	0.2959	0.0741	0.1618	-25.14%
w/o *SLGCN*	0.1466	0.5277	0.3149	0.1061	0.1361	-30.50%



(a) F1-score results (Vary k, $d = 5$) (b) F1-score results (Vary d, $k = 15$) (c) Efficiency results (Vary k, $d = 5$) (d) Efficiency results (Vary d, $k = 15$)

Figure 11: Effectiveness and Efficiency evaluation on GenCAT graph

well on 3-hop. This indicates that the relevant structural and attribute-based features for defining skyline communities are typically contained within a neighborhood of each node.

Exp-10: Varying epoch numbers. In Figure 10(d), we evaluate the effect of training epochs during the pre-training phase. On small graphs like Slashdot and Delicious, the performance increases as the number of epochs increases. In contrast, large graphs such as Lastfm and Flixster exhibit similar performance after 50 epochs, indicating that the model convergence is achieved within the first 50 epochs.

6.5 Additional Experiments

Exp-11: Ablation study with independent attributes ($d = 5$, $k = 15$). In this experiment, we evaluate the effectiveness of the components of DeepSCS, including the dominance loss ($\mathcal{L}_d$), the link loss ($\mathcal{L}_k$), and the overall architecture of SLGCN. The results is summarized in Table 4. The dominance loss ($\mathcal{L}_d$) plays an important role for large graphs. For example, on Flixster, our method can obtain a remarkable performance improvement of 80.3% when equipped with $\mathcal{L}_d$. In general, it contributes to an average of 29.61% F1-score improvement. Similarly, link loss ($\mathcal{L}_k$) also demonstrates great performance with an average of 25.14% improvement for F1-score. To further assess the SLGCN architecture, we replaced it with a classical contrast-based self-supervised approach for node representations learning. The results reveal that SLGCN achieves an average improvement of 30.5%. These results collectively demonstrate the effectiveness of the modules designed in DeepSCS.

Exp-12: Experiments on synthetic dataset GenCAT. Figure 11 reports the experiment results on synthetic dataset GenCAT. The F1-score outcomes for vary parameters k and d are presented in Figure 11(a) and (b), while Figure 11 (c) and (d) illustrate the efficiency results on GenCAT with independent attributes. These findings further validate the effectiveness and efficiency of DeepSCS.

7 RELATED WORKS

Traditional community search. Traditional CS methods aim to identify cohesively connected subgraphs that contain specific query nodes and satisfy given constraints. Many community models have been proposed, including k-core [8][30][31], k-truss [2][14][33], and maximal k-edge connected subgraphs [3][13][43]. However, these community models only consider the graph structural information and ignore the attributes associated with nodes. Li et al. introduced an influential community model [21] and a skyline community model [20], which considers the influence of nodes and captures d-dimensional numerical attributes, respectively. However, these approaches have two limitations: structural inflexibility and computational inefficiency.

Deep learning based community search. Recently, there has been increasing interest in learning-based community search (CS) methods. QD-GNN and AQD-GNN were proposed in [15] for CS and attributed community search in a supervised manner. TransZero [34] was introduced for CS in an unsupervised manner, without utilizing labels for the nodes in the ground-truth community. [12] employs k-core information as labels for pre-training and predicts the k-core community, while it does not use ground-truth community information.

Multi-valued graph analysis. In many real-world networks, such as social networks, each node is characterized by multi-valued attributes. Recent research has introduced various methodologies to tackle the challenges of analyzing such networks. Techniques like graph clustering [22, 32, 40], classification [6, 18], and subgraph matching [1] have become prominent in the fields of graph mining[27]. In this paper, we investigate the query-dependent skyline community search problem, and the proposed techniques are parallel to those in [12][34], but with more constraints arising from the properties of skyline community.

8 CONCLUSION

In this paper, we propose a deep unsupervised method for skyline community search. In the offline pre-training phase, we adapt graph convolutional networks to efficiently handle high-dimensional multi-valued graph, employing a dominance loss function to enhance community identification. In the online search phase, we calculate the skyline community score based on the learned representations, quantifying the similarity between query nodes and graph nodes. Extensive experiments demonstrate the efficiency, scalability, and effectiveness of our solutions.

ACKNOWLEDGMENT

This work was supported in part by the National Key R&D Program of China (2022YFB3103701), in part by the Major Key Project of PCL (PCL2024A05), in part by the Guangdong Basic and Applied Basic Research Foundation (202201020131 and 2023A1515011655), in part by the National Natural Science Foundation of China (U20B2046), and in part by the Australian Research Council (DP230101445 and FT210100303).

REFERENCES

[1] Shubhangi Agarwal, Sourav Dutta, and Arnab Bhattacharya. 2020. ChiSeL: graph similarity search using chi-squared statistics in large probabilistic graphs. *Proc. VLDB Endow.* 13, 10 (June 2020), 1654–1668. https://doi.org/10.14778/3401960.3401964

[2] Esra Akbas and Peixiang Zhao. 2017. Truss-based community search: a truss-equivalence based indexing approach. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1298–1309. https://doi.org/10.14778/3137628.3137640

[3] Uri Alon and Eran Yahav. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*. https://openreview.net/forum?id=i80OPhOCVH2

[4] Muhammad Attique, Muhammad Afzal, Farman Ali, Irfan Mehmood, Muhammad Fazal Ijaz, and Hyung-Ju Cho. 2020. Geo-Social Top-k and Skyline Keyword Queries on Road Networks. *Sensors (Basel, Switzerland)* 20 (2020).

[5] Mei Bai, Yuting Tan, Xite Wang, Bin Zhu, and Guanyu Li. 2021. Optimized Algorithm for Skyline Community Discovery in Multi-Valued Networks. *IEEE Access* 9 (2021), 37574–37589. https://doi.org/10.1109/ACCESS.2021.3063317

[6] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. 2011. *Node Classification in Social Networks*. Springer US, Boston, MA, 115–148. https://doi.org/10.1007/978-1-4419-8462-3_5

[7] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The Skyline operator. *Proceedings 17th International Conference on Data Engineering* (2001), 421–430.

[8] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (2014).

[9] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. 2016. Effective community search for large attributed graphs. *Proc. VLDB Endow.* 9, 12 (Aug. 2016), 1233–1244. https://doi.org/10.14778/2994509.2994538

[10] Yixiang Fang, Xin Huang, Lu Qin, Y. Zhang, W. Zhang, Reynold Cheng, and Xuemin Lin. 2019. A survey of community search over big graphs. *The VLDB Journal* 29 (2019), 353 – 392.

[11] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-Scale Learnable Graph Convolutional Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1416–1424. https://doi.org/10.1145/3219819.3219947

[12] Xiaoxuan Gou, Xiaoliang Xu, Xiangying Wu, Runhuai Chen, Yuxiang Wang 0001, Tianxing Wu 0001, and Xiangyu Ke. 2023. Effective and Efficient Community Search with Graph Embeddings. In *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023) (Frontiers in Artificial Intelligence and Applications)*, Kobi Gal, Ann Nowé, Grzegorz J. Nalepa, Roy Fairstein, and Roxana Radulescu (Eds.), Vol. 372. IOS Press, 891–898. https://doi.org/10.3233/FAIA230358

[13] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2016. Querying Minimal Steiner Maximum-Connected Subgraphs in Large Graphs. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 1241–1250. https://doi.org/10.1145/2983323.2983748

[14] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. Association for Computing Machinery, New York, NY, USA, 1311–1322. https://doi.org/10.1145/2588555.2610495

[15] Yuli Jiang, Yu Rong, Hong Cheng, Xin Huang, Kangfei Zhao, and Junzhou Huang. 2022. Query driven-graph neural networks for community search: from non-attributed, attributed, to interactive attributed. *Proceedings of the VLDB Endowment* 15 (02 2022), 1243–1255. https://doi.org/10.14778/3514061.3514070

[16] Junghoon Kim, Siqiang Luo, Gao Cong, and Wenyuan Yu. 2022. DMCS: Density Modularity based Community Search. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 889–903. https://doi.org/10.1145/3514221.3526137

[17] Thomas Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv* abs/1609.02907 (2016).

[18] Krishna Kumar P., Paul Langton, and Wolfgang Gatterbauer. 2020. Factorized Graph Representations for Semi-Supervised Learning from Sparse Data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1383–1398. https://doi.org/10.1145/3318464.3380577

[19] Ling Li, Siqiang Luo, Yuhai Zhao, Caihua Shan, Zhengkui Wang, and Lu Qin. 2023. COCLEP: Contrastive Learning-based Semi-Supervised Community Search. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. 2483–2495. https://doi.org/10.1109/ICDE55515.2023.00191

[20] Rong-Hua Li, Lu Qin, Fanghua Ye, Jeffrey Xu Yu, Xiaokui Xiao, Nong Xiao, and Zibin Zheng. 2018. Skyline Community Search in Multi-valued Networks. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18)*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3183713.3183736

[21] Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. 2015. Influential community search in large networks. *Proc. VLDB Endow.* 8, 5 (Jan. 2015), 509–520. https://doi.org/10.14778/2735479.2735484

[22] Mengqing Luo and Hui Yan. 2020. Adaptive Attributed Network Embedding for Community Detection. In *Pattern Recognition and Computer Vision: Third Chinese Conference, PRCV 2020, Nanjing, China, October 16–18, 2020, Proceedings, Part III*. Springer-Verlag, Berlin, Heidelberg, 161–172. https://doi.org/10.1007/978-3-030-60636-7_14

[23] Seiji Maekawa, Yuya Sasaki, George Fletcher, and Makoto Onizuka. 2023. GenCAT: Generating attributed graphs with controlled relationships between classes, attributes, and topology. *Inf. Syst.* 115, C (May 2023), 17. https://doi.org/10.1016/j.is.2023.102195

[24] Denis Mindolin and Jan Chomicki. 2009. Discovering relative importance of skyline attributes. *Proc. VLDB Endow.* 2, 1 (Aug. 2009), 610–621. https://doi.org/10.14778/1687627.1687697

[25] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2003. An optimal and progressive algorithm for skyline queries. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*. Association for Computing Machinery, New York, NY, USA, 467–478. https://doi.org/10.1145/872757.872814

[26] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. 2007. Probabilistic skylines on uncertain data. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07)*. VLDB Endowment, 15–26.

[27] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. 2017. The ubiquity of large graphs and surprising challenges of graph processing. *Proc. VLDB Endow.* 11, 4 (Dec. 2017), 420–431. https://doi.org/10.1145/3186728.3164139

[28] Yutaka Sasaki et al. 2007. The truth of the f-measure. 2007. 49 (2007).

[29] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 815–823. https://doi.org/10.1109/CVPR.2015.7298682

[30] Stephen B. Seidman. 1983. Network structure and minimum degree. *Social Networks* 5, 3 (1983), 269–287. https://doi.org/10.1016/0378-8733(83)90028-X

[31] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*. Association for Computing Machinery, New York, NY, USA, 939–948. https://doi.org/10.1145/1835804.1835923

[32] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed graph clustering: a deep attentional embedding approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*. AAAI Press, 3670–3676.

[33] Jia Wang and James Cheng. 2012. Truss decomposition in massive networks. *Proc. VLDB Endow.* 5, 9 (May 2012), 812–823. https://doi.org/10.14778/2311906.2311909

[34] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Efficient Unsupervised Community Search with Pre-Trained Graph Transformer. *Proc. VLDB Endow.* 17, 9 (Aug. 2024), 2227–2240. https://doi.org/10.14778/3665844.3665853

[35] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Neural Attributed Community Search at Billion Scale. *Proc. ACM Manag. Data* 1, 4, Article 251 (April 2024), 25 pages. https://doi.org/10.1145/3626738

[36] Jianwei Wang, Ying Zhang, Kai Wang, Xuemin Lin, and Wenjie Zhang. 2024. Missing Data Imputation with Uncertainty-Driven Network. *Proc. ACM Manag. Data* 2, 3, Article 117 (May 2024), 25 pages. https://doi.org/10.1145/3654920

[37] Xueyi Wu, Yuanyuan Xu, Wenjie Zhang, and Ying Zhang. 2023. Billion-scale bipartite graph embedding: A global-local induced approach. *Proceedings of the VLDB Endowment* 17, 2 (2023), 175–183.

[38] Yuanyuan Xu, Wenjie Zhang, Xiwei Xu, Binghao Li, and Ying Zhang. 2024. Scalable and Effective Temporal Graph Representation Learning with Hyperbolic Geometry. *IEEE Trans. Neural Networks Learn. Syst.* (2024), 2162–237X.

[39] Yuanyuan Xu, Wenjie Zhang, Ying Zhang, Maria E. Orlowska, and Xuemin Lin. 2024. TimeSGN: Scalable and Effective Temporal Graph Neural Network. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 3297–3310. https://doi.org/10.1109/ICDE60146.2024.00255

[40] Fanghua Ye, Chuan Chen, and Zibin Zheng. 2018. Deep Autoencoder-like Nonnegative Matrix Factorization for Community Detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 1393–1402. https://doi.org/10.1145/3269206.3271697

[41] Yutong Ye, Xiang Lian, and Mingsong Chen. 2024. Efficient Exact Subgraph Matching via GNN-Based Path Dominance Embedding. *Proc. VLDB Endow.* 17, 7 (May 2024), 1628–1641. https://doi.org/10.14778/3654621.3654630

[42] Dongxiao Yu, Lifang Zhang, Qi Luo, Xiuzhen Cheng, Jiguo Yu, and Zhipeng Cai. 2020. Fast skyline community search in multi-valued networks. *Big Data Mining and Analytics* 3, 3 (2020), 171–180. https://doi.org/10.26599/BDMA.2020.9020002

[43] Rui Zhou, Chengfei Liu, Jeffrey Xu Yu, Weifa Liang, Baichen Chen, and Jianxin Li. 2012. Finding maximal k-edge-connected subgraphs from a large graph. In *International Conference on Extending Database Technology*. https://api.semanticscholar.org/CorpusID:13377284