Skyline Community Search over Edge-Attributed Bipartite Graphs

Anonymous Author(s)

Abstract. Bipartite graphs, modeling relationships between two types of entities, are widely used in practical applications. Community search, a fundamental problem in bipartite graphs, has gained significant attention. However, existing studies focus on measuring structural cohesiveness between vertex sets while either ignoring edge attributes or considering only one-dimensional importance. In this paper, we introduce a novel community model, named edge-attributed skyline community (ESC), which preserves structural cohesiveness and captures the inherent dominance of multi-dimensional edge attributes in bipartite graphs. To search for ESCs, we developed an efficient peeling algorithm that iteratively deletes edges with the minimum attribute in each dimension. Additionally, we devised an expanding algorithm to reduce the search space and speed up the filtering of unpromising vertices using a proven upper bound. Extensive experiments on large-scale real-world datasets demonstrate the efficiency, effectiveness, and scalability of our approach. A case study compared with prior arts demonstrates that our design improves the precision and diversity of results.

Keywords: Bipartite graph · cohesive subgraph · community search.

1 Introduction

Numerous real-world networks consist of community structures, making their discovery crucial for network analysis [1, 36, 37]. Community search, a query-dependent discovery problem, aims to find the densest connected subgraphs containing specific input queries. With the increasing number of applications, it has garnered significant attention from database professionals in recent years [15, 32, 6, 35, 34, 25].

Traditional community search focuses primarily on the network's topology. However, in practical applications, it is crucial to consider not only the connections between entities but also their attribute characteristics, as these can enhance the cohesion of identified communities [14, 16]. This makes community search over attributed graphs particularly significant. Most current research addresses community constraints in terms of structure and attributes but typically only for scenarios involving a single type of entity [24, 13]. However, in scenarios involving two different types of entities, such as collaboration networks [18], customer-product networks [27, 33], and user-location networks [5], the focus tends to be on structural cohesiveness, often neglecting attributes. Approaches

like the (α, β)-core [8], bitruss [26, 28], and biclique [39, 29] exemplify this trend by primarily concentrating on structure.

Some researchers considered one-dimensional importance for attribute information [31], but interactions in a graph often involve multiple dimensions. These methods miss the multidimensional aspects and fail to capture all cohesive communities. We aim to develop a model using both structural and multidimensional attribute information to identify all cohesive communities containing the queries. As depicted in Fig 1, the IMDB network's bipartite graph includes relationships between directors and movies. The table illustrates directors' multidimensional evaluation attributes for movies. By designating U_2 as the query vertex and setting structural constraints of $\alpha = \beta = 2$, we can extract two mutually exclusive communities with the largest connected subgraphs based on the multidimensional attributes.

In this study, we aim to develop a method considering the graph's structural information, edge attribute information, and cohesiveness. Our goal is to identify vertices in the bipartite graph that are structurally closest to the query vertex and have similar attribute values, along with the corresponding edges, to find high-importance communities that meet these criteria. To achieve this, we propose a skyline community search problem tailored to bipartite graphs with multidimensional edge attributes, named Edge-attributed Skyline Community Search Problem **(ESCS-Problem)**, along with its model termed Edge-attributed Skyline Community **(ESC)**.

Our initial effort targets bipartite graphs with multidimensional edge attributes, presenting a new challenge. Previous research doesn't fully address our specific application. A recent study [20] focuses on communities with multidimensional attributes in simple graphs, while another [30] deals with single-dimensional attributes, making it a subset of our scope. Additionally, the design [40] using Pareto optimality for one-dimensional weights in bipartite graphs differs in context and approach. Other works [41, 23] on attribute community discovery in heterogeneous graphs handle only one-dimensional attributes and vertex-associated values, unsuitable for our scenario. We face challenges in balancing structural compactness with multidimensional attribute advantages, marking a creative endeavor. Our contributions are summarized as follows.

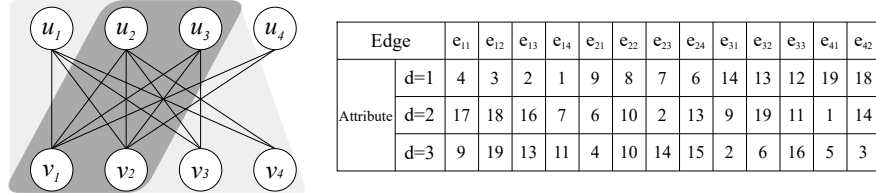| Edge		e_{11}	e_{12}	e_{13}	e_{14}	e_{21}	e_{22}	e_{23}	e_{24}	e_{31}	e_{32}	e_{33}	e_{41}	e_{42}
	d=1	4	3	2	1	9	8	7	6	14	13	12	19	18
Attribute	d=2	17	18	16	7	6	10	2	13	9	19	11	1	14
	d=3	9	19	13	11	4	10	14	15	2	6	16	5	3

Fig. 1: Example of localized IMDB bipartite graph.

- **Novel community model.** We introduce a new model for community search in bipartite graphs, considering both structural cohesion and multi-dimensional edge attributes.
- **New algorithms.** We propose an efficient peeling algorithm that removes edges with the minimum attribute and an expanding algorithm to reduce search space and speed up filtering. We also give lemmas to improve search performance, and verify computational complexity and completeness.
- **Extensive experiments.** We conduct extensive experiments over six real-world datasets to demonstrate the high efficiency and scalability of our proposed algorithms. We also give an extensive case study, demonstrating our ESC model's capability to match dense subgraphs of bipartite graphs with greater accuracy and personalization.

Extensive Applications for ESC:

(1) **Personalized Recommendation Systems.** Our ESC model surpasses current solutions by integrating structural cohesion and multi-dimensional edge attributes. Unlike existing models that are constrained to either structure or single attributes, ESC captures a diverse range of user behaviors. For movie recommendations, it considers ratings, likes, and viewing duration. On social platforms, it refines blogger recommendations by simultaneously analyzing likes, follows, and comments.

(2) **Team Formation.** The ESC model enhances team assembly by balancing structural cohesion and multi-dimensional contributions. Unlike models that focus solely on experience, ESC evaluates factors such as impact, involvement duration, and deliverable volume to form teams with both deep expertise and relevant project experience. It ensures that the teams meet multiple objectives without making trade-offs, which is often a limitation of existing methods that may overlook the importance of maintaining such a balance.

(3) **Anomaly Detection.** The ESC model enhances fraud detection by utilizing multi-dimensional bipartite graphs, surpassing models that rely on single anomaly indicators. By jointly assessing factors such as financial scale, affected users, and severity, it ensures robust risk identification. Its ability to detect diverse yet structurally significant anomalies strengthens fraud prevention, addressing gaps left by traditional models.

2   Problem Statement

Definition 1. (Bipartite Graph.) *A bipartite graph $G = (U, L, E, X)$ is a graph whose vertices can be divided into two disjoint sets such that every edge connects a vertex from one set to a vertex from the other set. Note that U and L denote the sets of upper and lower vertices, respectively, and E represents the set of edges in G. The edges in G have d numerical attributes, which are denoted by X_i, where $X_i \in X$ and X_i is a d-dimensional vector. Let $|*|$ denote the modulo operation. We have $|U| + |L| = n$ and $|E| = m$.*

Definition 2. ((α, β)-core.) *Given a bipartite graph $G = (U, L, E, X)$, an (α, β)-core is a connected subgraph where the degree of vertices in the upper layer is at least α, and the degree of vertices in the lower layer is at least β.*

Definition 3. (Significance.) *Given a bipartite graph $G = (U, L, E, X)$ with edge attributes, we define the significance of G on the ith dimension (for $i = 1, 2, \cdots, d$) as: $f_i(G) = \min_{e \in |E|} X_i^e$.*

Definition 4. (Domination.) *Given two bipartite graphs $G = (U, L, E, X)$ and $G' = (U', L', E', X')$, if $f_i(G) \leq f_i(G')$ (for $i = 1, 2, \cdots, d$) and there exists $f_i(G) < f_i(G')$ for certain i, we call G' dominates G, denoted by $G \preceq G'$.*

The significance bounds an attribute dimension, with higher values indicating stronger cohesion. In a user-movie network, edge attributes represent ratings, similar ratings suggesting community membership. The final output subgraph, ESC, must not be dominated-any lower significance implies non-optimality.

Definition 5. (ESC.) *Given a bipartite graph $G = (U, L, E, X)$, α, β and query vertex q, where X is the multi-dimensional significance on edges [20, ?, ?]. An ESC is a connected subgraph $H = (U_H, L_H, E_H, X_H)$ of G such that it satisfies:*

1) Cohesive property: *H is an (α , β)-core contains q. It ensures that the connected subgraph contains the query and follows the (α, β)-core structure.*
2) Skyline property: *it does not exist a subgraph H' of G such that H' is an (α, β)-core and $H \prec H'$. It guarantees that the subgraph is not dominated by others, making it an optimal solution.*
3) Maximal property: *it does not exist a subgraph H' of G such that (1) H' is an (α, β)-core, (2) H' contains H, and (3) $f_i(H) = f_i(H')$ for all $i \in [1, d]$. It ensures the largest possible subgraph under constraints, maximizing member inclusion.*

ESCS-Problem: Given a bipartite graph G, a query vertex q, and parameters α and β for structural constraints, the problem is aimed to search all ESCs containing q, where each ESC H cannot be dominated by any other ESC H'.

Example 1. To illustrate the definition of ESC, Figure 1 depicts a bipartite graph segment from the IMDB network with three-dimensional edge attributes. Here, "e_{ij}" signifies the edge that connects node u_i with node v_j. We concentrate on two-dimensional attributes for simplicity. Let U_2 be the query vertex and $\alpha = \beta = 2$, leading to the identification of two ESCs, H_1 and H_2, each highlighted within a dotted circle. H_1 is composed of the vertices U_2, U_3, V_1, V_2, and H_2 includes $U_1, U_2, U_3, V_1, V_2, V_3, V_4$. Both H_1 and H_2 incorporate U_2 and are defined as (2, 2)-cores. The significance of H_1 is captured by $f(H_1) = (8, 6)$, and for H_2, it is $f(H_2) = (1, 7)$. This configuration ensures that they are impervious to dominance by other (2, 2)-cores based on their significance.

3 Peeling-based Algorithm

We define the ESC and propose algorithms to detect it in large-scale bipartite graphs by combining (α, β)-core constraints with significance measures. The size gap between the ESC and the original graph impacts efficiency, so we introduce two approaches: a peeling algorithm for large disparities and an expanding algorithm for smaller ones. The peeling method iteratively removes edges that violate conditions, making it efficient when the ESC is much smaller. Prioritizing edge removal based on multidimensional significance, our approach adapts to different dimensions, recognizing their hierarchical structure for optimal processing.

3.1 Peeling for $d = 1$

In simple bipartite graphs with one-dimensional attributes, the algorithm follows a straightforward approach. It first sorts edges in ascending order of significance, then iteratively removes the least significant edge until the structure no longer maintains an (α, β)-core or the queried vertex is excluded. The resulting (α, β)-core is called an ESC. In the operational phase, peeling one edge at a time is inefficient. To address this, we've introduced DFSCom, as depicted in Alg. 1, which recursively traverses adjacent vertices to ensure all structural constraints are met. Once these criteria are satisfied, the algorithm ends, removing all non-compliant vertices and their edges. It significantly reduces the algorithm's iteration count.

Completeness: The integrity of the algorithm is elucidated as follows. Assuming there exists another ESC H' that contains edges not present in the current ESC H. It must satisfy at least one of the following two conditions: (1) it contains edges with significance smaller than $f(H)$, which contradicts the definition of $f(H')$ and is not possible; or (2) it contains edges with significance greater than $f(H)$, which were removed because the two vertices they connect do not satisfy the structural constraints. When H' adds these edges to the community, it must also add edges with significance smaller than $f(H)$ in order to satisfy the (α, β)-core constraint, which contradicts the definition of $f(H')$ and is not possible. Therefore, it is not possible to have another ESC.

3.2 Peeling for $d = 2$

As edge significance dimensionality grows, a backtracking-based dimensionality reduction strategy decomposes two-dimensional significance into one-dimensional sets, enabling existing methods to tackle new problems.

Lemma 1. *Without loss of generality, assuming that there exist n ESCs H_i, $(i = 1, 2, ..., n)$ in the bipartite graph G with two dimensions. Then, the significance of each dimension corresponding to different ESCs can be related as:*

$$
\begin{aligned}
f_1(H_1) < f_1(H_2) \quad &< ... < f_1(H_i) \quad < ... < f_1(H_n) \\
f_2(H_1) > f_2(H_2) \quad &> ... > f_2(H_i) \quad > ... > f_2(H_n)
\end{aligned}
\tag{1}
$$

Algorithm 1: PDim1

Input: $G, \alpha, \beta, q, I, d, F$

Output: f_d

1 $G \leftarrow G \setminus \{e | e \text{ violates } I\}$; $G(q) \leftarrow$ maximal (α, β)-core containing q;
$f_d \leftarrow f_d(G(q))$;

2 **while** $G(q) \neq \emptyset$ **do**

3 $\quad$ $w_{min} \leftarrow \min_{d\text{-dim}} G(q)$; **foreach** $(u, v) \in G(q)$ **do**

4 $\quad\quad$ $visited(u, v) \leftarrow 0$;

5 $\quad$ **foreach** $(u, v) \in G(q)$ *with* $w(u, v) = w_{min}$ **do**

6 $\quad\quad$ **if** $(u, v) \in F$ **then break**;

7 $\quad\quad$ $S \leftarrow \emptyset, visited(u, v) \leftarrow 1$;

8 $\quad\quad$ **if** $DFSCom(u, q, \alpha, \beta, S, F) = 0$ **then break**;

9 $\quad\quad$ **if** $DFSCom(v, q, \beta, \alpha, S, F) = 0$ **then break**;

10 $\quad\quad$ $G(q) \leftarrow G(q) \setminus \{(u, v)\}$; $f_d \leftarrow \max(f_d, f_d(G(q)))$;

11 **return** f_d;

12 **Function** DFSCom$(u, q, \alpha, \beta, S, F)$:

13 $\quad$ **if** $u = q \wedge deg(u, G(q)) = \alpha$ **then return** *0*;

14 $\quad$ Update $deg(u, G(q))$;

15 $\quad$ **if** $deg(u, G(q)) < \alpha$ **then**

16 $\quad\quad$ **foreach** $v' \in N(u)$ *with* $visited(u, v') = 0$ **do**

17 $\quad\quad\quad$ $visited(u, v') \leftarrow 1$; $S \leftarrow S \cup \{(u, v')\}$; **if**
$(u, v') \in F \vee DFSCom(v', q, \beta, \alpha, S, F) = 0$ **then**
$G(q) \leftarrow G(q) \cup S$; **return** *0* ;

18 $\quad\quad$ $S \leftarrow \emptyset$;

19 $\quad$ **return** *1*;

Algorithm 2: PDim2

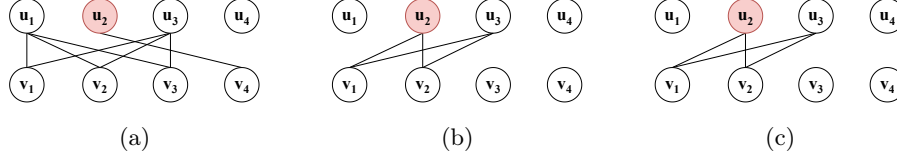
Input: $G, \alpha, \beta, q, I, F$

Output: R

1 $f_2 \leftarrow$ PDim1$(G, \alpha, \beta, q, I, 2, F)$;

2 $R \leftarrow \varnothing$;

3 **while** $f_2 > 0$ **do**

4 $\quad$ $I' \leftarrow I.update(\{X_2^i \geq f_2\})$;

5 $\quad$ $f_1 \leftarrow$ PDim1$(G, \alpha, \beta, q, I', 1, F)$;

6 $\quad$ $R \leftarrow R \cup \{(f_1, f_2)\}$;

7 $\quad$ $I' \leftarrow I.update(\{X_1^i > f_1\})$;

8 $\quad$ $f_2 \leftarrow$ PDim1$(G, \alpha, \beta, q, I', 2, F)$;

9 **return** R;

Fig. 2: An example of the peeling algorithm for $d = 2$.

Based on Lemma 1, we propose the Alg. 2.

Completeness: Suppose an undiscovered ESC H. According to Lemma 3, we have $f_1(H_i) < f_1(H) < f_1(H_{i+1})$ and $f_2(H_j) > f_2(H) > f_2(H_{j+1})$. Assuming $i < j$, we have $f_1(H_i) < f_1(H) < f_1(H_j)$. Also, since $f_2(H_j) > f_2(H)$, H_j dominates H, thus H cannot exist. The same can be proven when $i > j$. Therefore, Algorithm expanding($d = 2$) is complete.

Example 2. Referring to Figure 1, it is invoked on the significance of the second dimension to obtain an upper bound of $f_2(H_1)$. Then, all edges in the original graph G with $X_2^e < f_2(H_1)$ are removed. Algorithm expending (d=1) is applied to the attribute values of the first dimension, yielding $f_1(H_1) = 1$, resulting in the first $H_1 = (1, 7)$ as depicted in Fig 2(a). All edges in the original graph G with $X_1^e > f_1(H_1)$ are retained, as shown in Fig 2(b). Algorithm expending (d=1) is called upon again to update $f_2(H_2)$. This iterative process continues until $f_2 = 0$. At this point, as shown in Fig 2(c), another ESC $H_2 = (8, 6)$ is obtained.

3.3   Peeling for $d = 3$

As the attribute dimensionality on edges advances to 3, dimensionality reduction through backtracking continues to be relevant. Considering that Lemma 3 is not adaptable in a parallel fashion, alterations in the procedural specifics become imperative.

As shown in Alg. 3 and 4, we implement an alternate Algorithm expending ($d = 3$). For obtaining $f_3(H_i)$, the process of being transferred function CandSig is as follows: in the process of gradually removing the edge with the minimum X_d, recording each X_d that makes the subgraph contain (α,β)-core. These recorded values constitute all possible significance for ESCs on d_{th} dimension. Then, we iterate through the set V of all possible values $v_i \in V$ during each iteration, edges with third-dimensional attribute values $X_3^e \leq v_i$ are removed from the entire graph.

Completeness: Since Algorithm expending(d=1) achieves its goal through step-by-step removing edge, obtaining all possible $f_3(H_i)$ is exhaustive. If there were any other $f_3(H_i)$ present, it would be impossible to form an (α, β)-core. Based on this, a proof by contradiction is employed. When fixing the $f_3(H_i)$, assume the existence of an ESC H' in the subgraph on the other two dimensions that Algorithm expending(d=2) did not find. Since the completeness of Algorithm 2 has been established, H' must be dominated by the already identified

---

Algorithm 3: CandSig

Input: G, α, β, q, d
Output: T_d

1 $T_d \leftarrow \emptyset$; $G(q) \leftarrow$ maximal (α, β)-core containing q;
2 **while** $G(q) \neq \emptyset$ **do**
3 $f_d \leftarrow \min_{d\text{-dim}} G(q)$; $T_d \leftarrow T_d \cup \{f_d\}$;
4   Remove $e = (u, v) \in G(q)$ with $X_d^e = f_d$;
5   DFS($G(q)$, u, α, β); DFS($G(q)$, v, β, α);
6 $G(q) \leftarrow$ maximal (α, β)-core containing q;

7 **return** T_d;
8 **Function** DFS(G, u, α, β)**:**
9   **if** $deg(G, u) > \alpha$ **then**
10   **foreach** $v \in N(u)$ **do**
11 $G \leftarrow G \setminus \{(u, v)\}$; DFS($G, v, \beta, \alpha$);

Algorithm 4: PDim3

Input: $G, \alpha, \beta, q, I, F$
Output: R

1 $F_3 \leftarrow$ CandSig($G, \alpha, \beta, q, 3$); $R, S \leftarrow \emptyset$; Sort F_3 descending;
2 **foreach** $f_3 \in F_3$ **do**
3 $e \leftarrow \{e \in G \mid X_3^e = f_3\}$; $F \leftarrow F \cup e$;
4   **if** $(X_1^e, X_2^e) \preceq S$ **then continue**;
5 $I' \leftarrow I \cup \{X_3 \geq f_3\}$; $T \leftarrow$ PDim2($G, \alpha, \beta, q, I', F$);
6 $S \leftarrow S \cup T$; **foreach** $(f_1, f_2) \in T$ **do**
7 $R \leftarrow R \cup \{(f_1, f_2, f_3)\}$;

8 **return** R;

ESCs, contradicting the definition of ESC. Therefore, the algorithm does not overlook ESC solutions, and the completeness of the algorithm is established.

3.4 Universal peeling

For high-dimensional cases, as shown in Alg. 5, we propose expanding $(d > 3)$ algorithm based on dimensionality reduction and backtracking. It collects all possible values of $f_d(H_i)$ as F_d and iterates over each f_d in F_d.

Algorithm 5: UniPDim

Input: $G, \alpha, \beta, q, I, F, d$
Output: R

1 **if** $d = 3$ **then return** $PDim3(G, \alpha, \beta, q, I, F)$;
2 $F_d \leftarrow \text{CandSig}(G, \alpha, \beta, q, d)$; $R, S \leftarrow \emptyset$; Sort F_d descending; **foreach** $f_d \in F_d$ **do**
3 $\quad$ $e \leftarrow \{e \in G \mid X_d^e = f_d\}$; $F \leftarrow F \cup e$;
4 $\quad$ **if** $(X_1^e, \ldots, X_{d-1}^e) \preceq S$ **then** Continue;
5 $\quad$ $I' \leftarrow I \cup \{X_d \geq f_d\}$; $T \leftarrow \text{UniPDim}(G, \alpha, \beta, q, I', F, d-1)$;
6 $\quad$ $S \leftarrow S \cup T$;
7 $\quad$ **foreach** $(f_1, \ldots, f_{d-1}) \in T$ **do**
8 $\quad\quad$ $R \leftarrow R \cup \{(f_1, \ldots, f_{d-1}, f_d)\}$;

9 **return** R;

In each iteration, edges with d_{th} dimensional significance $X_d^e \leq f_d$ are removed, and the $(d-1)_{th}$ peeling algorithm is applied, reducing dimensionality until $d = 3$. Then, it retrieves all relevant ESCs. A pruning strategy improves efficiency by preventing redundant calculations. For each ESC, f_d ensures that edge e with $X_d^e = f_d$ is present, setting upper bounds on other dimensions based on e's significance. If these bounds are dominated by previously found ESCs, no ESC corresponds to f_d, accelerating computation. We further introduce PDim1, PDim2, and PDim3 for low-dimensional ESC search, simplifying dimensionality reduction. Additionally, UniPDim offers a universal approach for any $d > 1$, resembling mathematical induction.

4 Expanding-based Algorithm

The peeling-based algorithm is proficient and effective in handling large-scale graphs, albeit with intrinsic limitations. Intuitively, each search requires loading the entire graph to execute the peeling process, which becomes excessively resource-intensive when dealing with smaller graph scales-transitioning from a large graph to a significantly smaller one. To address this, we introduce a multidimensional approach utilizing edge expanding.

4.1   Expanding for $d = 1$

As shown in Alg. 6, we tailor for the scenario where dimensionality $d = 1$. A straightforward method is to sort the edges of the initial graph by their significance. The iteration involves progressively incorporating the edge with the maximal value into an empty graph until it evolves into an (α,β)-core encompassing all queries. However, its efficiency is curtailed due to the addition of merely one edge per iteration. To mitigate this, we introduce Lemma 2 to establish a pseudo-upper bound for f_1 within the ESC. Leveraging this pseudo-upper bound allows for batch processing of edges, thereby substantially elevating the efficiency of edge integration.

Algorithm 6: EDim1

Input: G, α, β, q, d
Output: R

1 $G^*, C^*, R \leftarrow \varnothing$, $count \leftarrow \alpha$ or β; $f \leftarrow count^{th}$-max X_1^i of q's edges;
2 **while** $|E(G)| \geq |E(G^*)|$ **do**
3 $G^* \leftarrow \{e | X_1^i > f\}$; $C^* \leftarrow$ connected component in G^*;
4   **if** C^* *changed* $\wedge$ C^* *satisfies Lemmas 3 & 4* **then**
5   **if** $q \in C^*$ **then**
6 $H \leftarrow (\alpha, \beta)$-core of C^*; **if** $q \in H$ **then**
7 $R \leftarrow H$; **break**;

8 $count \leftarrow count + 1$; $f \leftarrow count^{th}$-max X_1^i in G;
9 **return** R;

Completeness: Assuming the existence of an ESC H' with $f_1(H') \geq f_1(H)$, H' must belong to one of the following two cases: (1) it contains edges that are not in H, or (2) the edge set of H' is a subset of the edge set of H. Since the algorithm continuously adds edges to an empty graph, the resulting community is the first (α, β)-core that includes the query. If H' contains edges that are not in H, these edges must have significance less than $f_1(H)$, and if H' contains such edges, it contradicts the definition of $f_1(H')$. So, the first case does not exist. In the second case, if the edge set of H' is a subset of the edge set of H, then H' is not the largest (α,β)-core, contradicting the definition of ESC, so there are no other ESCs.

Lemma 2. *Given the query vertex q, the degree on the layer of q is α, the set of significance for edges connected to q store in N_E. The pseudo upper bound δ of the significance of the ESC is the αth maximal value in the sorted set N_E.*

By applying Lemma 2 to determine the pseudo upper bound δ, it used to incorporate only those edges where $X_1^e \geq \delta$, thereby markedly enhancing time efficiency. If the edges recently integrated fail to constitute an (α,β)-core encompassing the queries, the process necessitates continued iterations, sequentially

incorporating remaining edges with the maximum significance into the current graph. Furthermore, we integrate insights from Lemma 3 and Lemma 4 [30], to further curtail the number of iterations and amplify efficiency. This culminates in the introduction of Algorithm peeling(d=1), a novel approach specifically devised to tackle the ESC search problem.

Lemma 3. *If a connected subgraph C^* contains an ESC H, then there must be:*

$$\alpha\beta - \alpha - \beta \leq |E(C^*)| - |U(C^*)| - |L(C^*)| \tag{2}$$

Lemma 4. *If a connected subgraph C^* contains an ESC H, then C^* has at least β vertices of degree α and α vertices of degree β.*

4.2   Expanding for $d = 2$

When the edge significance dimension is set to 2, multiple ESCs may exist. Identifying all ESCs requires both edge addition and deletion. We employ a dimensionality reduction approach, processing the two significance dimensions separately. The initial ESC community H is derived using Algorithm expending($d = 1$) based on the second-dimensional attribute. By Lemma 1, $f_1(H)$ is the lower bound of the first-dimensional significance for all ESCs. A new ESC H' with $f_1(H') > f_1(H)$ is then found by removing edges with $X_1^e = f_1(H)$ and adding those with $X_1^e > f_1(H)$. This iterative process continues until all ESCs are identified, forming the basis of Algorithm expending ($d = 2$).

As shown in Alg. 7, it initializes key variables: G^* (accumulated edges), f_1 (current ESC's first-dimensional significance), R (set of ESCs), *count* (tracking edge order by second-dimensional significance), and D (operating dimension). ESC identification consists of two steps: finding the first ESC and generating subsequent ones.

S stores the (α, β)-core containing the query within G^*. The algorithm iteratively removes the edge with the smallest value in dimension D until the core condition no longer holds. If S dominates existing ESCs in R, they are removed; if S is empty, the process continues or terminates based on f_1. Otherwise, f_1 is updated, and the new ESC is added to R.

Example 3. Using the example in Fig 1 with query vertex U_2 and $\alpha = \beta = 2$, we apply Algorithm peeling(d=2) to find all ESCs, as shown in Fig 3. First, let f be the second-largest second-dimensional significance of edges connected to U_2, initially 10. Edges with $X_2^e > 10$ are added to G^* (upper part of Fig 3a). Updating f to 9, we continue adding edges until $f = 7$, forming the first ESC (1,7). The lower part of Fig 3a highlights the additional edges in red. Next, removing edge (u_1, v_4) and adding edges with $X_1^e > 1$ to G^*, we iteratively delete the edge with the smallest second-dimensional significance to obtain ESC (2,6) (Fig 3b). Repeating this process generates ESCs (3,6), (4,6), and (8,6), but due to dominance relationships, the final ESCs are (1,7) and (8,6) (Fig 3c).

Algorithm 7: EDim2

Input: G, α, β, q, F
Output: R

1 $G^*, R \leftarrow \varnothing$, $f_1 \leftarrow -1$, $count \leftarrow \alpha$ or β, $D \leftarrow 1$;
   $f \leftarrow count^{th}$-max X_2^i of q's edges;
2 while $|E(G^*)| < |E(G)|$ **do**
3   **if** $f_1 = -1$ **then**
4 | $G^* \leftarrow \{e | X_2^i > f\}$; $D \leftarrow 1$, $count \leftarrow count + 1$;
         | $f \leftarrow count^{th}$-max X_2^i in G;
5   **else**
6 | $G^* \leftarrow (G^* \setminus \{e | X_1^i = f_1\}) \cup \{e | X_1^i > f_1\}$; $D \leftarrow 2$;
7 $S \leftarrow \varnothing$; **while** $H \leftarrow (\alpha, \beta)$-core of $G^* \neq \emptyset$ and $q \in H$ and $F \subseteq H$ **do**
8 | $S \leftarrow H$; Remove edge with min X_D^i from G^*;
9   **if** $S = \varnothing$ **then**
10 | **if** $f_1 = -1$ **then** continue;
11 | **break**
12   **else if** $\exists ESC \in R \preceq S$ **then**
13 | Remove dominated ESC from R
14 $(f_1, f_2) \leftarrow S$; $R \leftarrow R \cup \{(f_1, f_2)\}$;
15 return R;

4.3   Expanding for $d = 3$

Expanding strategies can still be extended to three-dimensional situations. Utilizing a dimensionality reduction strategy, the values of the three-dimensional attributes are efficiently decomposed into two distinct sets: one set of one-dimensional values and another set of two-dimensional values. A preliminary and simplistic strategy entails enumerating all feasible values of f_3, setting each of these values consecutively, and subsequently conducting a subgraph analysis to locate ESCs within the context of the remaining two-dimensional attributes.

However, this method fails to guarantee finding ESCs for every f_3 value. Fixing f_3 constrains the search domain to $C_0 = \{X_1^e > 0, X_2^e > 0\}$, as shown in Figure 4a. An ESC H defines a rectangular region in the solution space (shaded area in Figure 4b). A new ESC within this region would be subordinate to H, leading to redundant computations and reduced efficiency.

To optimize the search, we use recursive decomposition, pruning unnecessary computations. An ESC must lie outside the shaded area, but its irregular shape prevents direct application of Algorithm peeling(d=2). Thus, we partition the solution space into $C_1 = \{X_1^e > 0, X_2^e > f_2(H)\}$ and $C_2 = \{X_1^e > f_1(H), X_2^e > 0\}$, enabling separate applications of Algorithm peeling(d=2). Each region's lower bounds are represented as $(0, f_2(H))$ for C_1 and $(f_1(H), 0)$ for C_2.
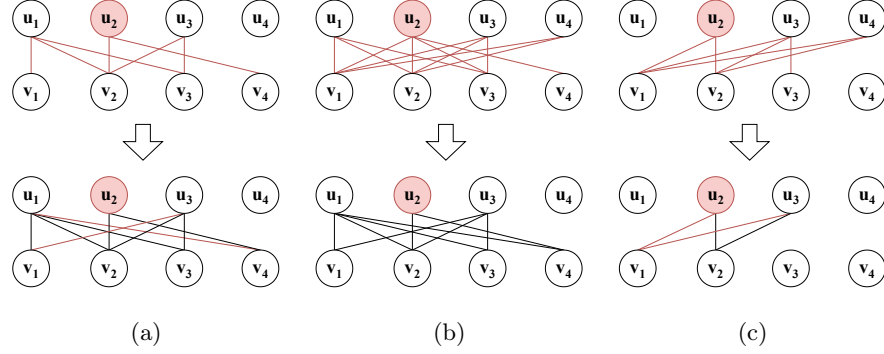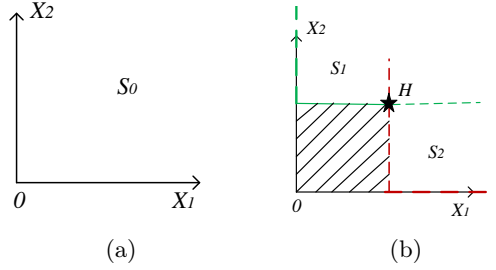
Fig. 3: An example of expanding in case $d = 2$.

Fig. 4: Recursive decomposition approach.

Lemma 5. *Suppose it exists n ESCs: $H_i = (f_1(H_i), f_2(H_i))$, $i \in [1, n]$, arranging all the horizontal and vertical coordinates as:*

$$
\begin{aligned}
0 &< f_1(H_1) < f_1(H_2) < ... < f_1(H_n) \\
f_2(H_1) &> f_2(H_2) > ... > f_2(H_n) > 0
\end{aligned}
\tag{3}
$$

By combining the coordinates of the upper and lower inequalities, the solution space is: $C_1 = (0, f_2(H_1)), C_2 = (f_1(H_1), f_2(H_2)),...,C_{n+1} = (f_1(H_n), 0)$.

As shown in Alg. 8, we combine the aforementioned recursive decomposition manner for solving the ESC search problem in three-dimensional attribute bipartite graphs, which initially creates an empty graph G^* for each solution space. Subsequently, it iteratively adds a batch of edges to G^* that meet the significance requirements of the solution space. The algorithm then calls Algorithm peeling(d=2) on G^* to compute ESCs. Following this, leveraging Lemma 5, it updates the solution space using the generated ESCs and calculates f_3 for each solution space. This process iterates until all ESCs are identified.

Algorithm 8: EDim3

Input: G, α, β, q, F
Output: R

1 $G^*, R, Q, P \leftarrow \varnothing$; $Q \leftarrow Q \cup (0, 0, \mathrm{EDim1}(G, \alpha, \beta, q, 3))$;
2 **while** $Q \neq \emptyset$ **do**
3 $f_3 \leftarrow \max_{\text{3-dim}} Q$; $S \leftarrow \varnothing$;
4   **foreach** $(p_1, p_2, f_3) \in Q$ **do**
5 $G^* \leftarrow \{e | x_1^e > p_1, x_2^e > p_2, x_3^e \geq f_3\}$; Remove (p_1, p_2, f_3) from Q; **if** $x_3^e = f_3$ **then** $F \leftarrow F \cup e$;
6 $S \leftarrow S \cup \mathrm{EDim2}(G^*, \alpha, \beta, q, F)$;
7   **foreach** $(f_1, f_2) \in S$ **do**
8   **if** $(f_1, f_2) \succeq P$ **then**
9   Remove dominated from P;
10   **else if** $(f_1, f_2) \preceq P$ **then**
11   continue;
12 $Q \leftarrow Q \cup (f_1, f_2, f_3)$; $P \leftarrow P \cup (f_1, f_2)$;
13   **foreach** $(point_1, point_2) \in Lemma5$ **do**
14 $G^* \leftarrow \{e | x_1^e > point_1, x_2^e > point_2\}$;
             $Q \leftarrow Q \cup (point_1, point_2, \mathrm{EDim1}(G^*, \alpha, \beta, q, 3))$;

15 return R;

4.4   Universal expanding

Dimensionality reduction can be used for high-dimensional attributes. As shown in Alg. 9, we propose Algorithm peeling $(d > 3)$ for the ESC search problem, following the methodology of Algorithm peeling $(d = 3)$ but extending the solution space update in recursive decomposition based on Lemma 5.

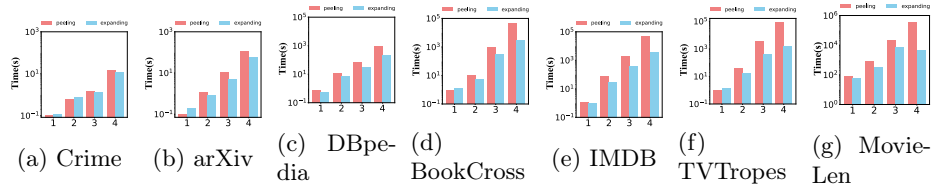(a) Crime   (b) arXiv   (c) DBpedia   (d) BookCross   (e) IMDB   (f) TVTropes   (g) MovieLen

Fig. 5: The efficiency of the algorithms w.r.t dimensions change.

5   Experiment

We validate the efficiency by varying the significance dimension d and the structural parameters α and β. To assess scalability, we change the dataset size.

Algorithm 9: UniEDim

Input: $G, \alpha, \beta, q, F, d$
Output: R

1 **if** $d = 3$ **then** return EDim3(G, α, β, q, F);
2 $G^*, R, Q, P \leftarrow \varnothing$; $Q \leftarrow Q \cup (0, \ldots, f_d^{max})$; $C \leftarrow \{(0, \ldots, 0)_{d-1}\}$; $f_d^{max} \leftarrow$
   EDim1(G, α, β, q, d);
3 **while** $Q \neq \emptyset$ **do**
4 $f_d \leftarrow \max_{d\text{-dim}} Q$; $S \leftarrow \varnothing$; **foreach** $(p_1, \ldots, p_{d-1}, f_d) \in Q$ **do**
5 $G^* \leftarrow \varnothing$; Add edge e with $\{x_1^e > p_1, \ldots, x_d^e \geq f_d\}$ to G^*; Remove
          $(p_1, \ldots, p_{d-1}, f_d)$ from Q; **if** $x_d^e = f_d$ **then** $F \leftarrow F \cup e$;
6 $S \leftarrow S \cup$ UniEDim$(G^*, \alpha, \beta, q, F, d - 1)$;
7   **foreach** $(f_1, \ldots, f_{d-1}) \in S$ **do**
8   **if** $(f_1, \ldots, f_{d-1}) \succeq P$ **then**
9   Remove dominated from P;
10   **else if** $(f_1, \ldots, f_{d-1}) \preceq P$ **then**
11   continue;
12 $Q \leftarrow Q \cup (f_1, \ldots, f_{d-1}, f_d)$; $P \leftarrow P \cup (f_1, \ldots, f_{d-1})$; $C \leftarrow$
             DivideSpace$(C, (f_1, \ldots, f_{d-1}))$;
13   **foreach** $(point_1, \ldots, point_{d-1}) \in C$ **do**
14 $G^* \leftarrow \varnothing$; Add edge e with $\{x_1^e > point_1, \ldots, x_{d-1}^e > point_{d-1}\}$ to
             G^*; $Q \leftarrow Q \cup (point_1, \ldots, point_{d-1}, \text{EDim1}(G^*, \alpha, \beta, q, d))$;

15 return R;
16 **Function** DivideSpace$(C, (f_1, \ldots, f_k))$:
17   **foreach** $i \in 1, \ldots, k$ **do**
18 $C' \leftarrow \varnothing$; **foreach** $c \in C$ *where* $(f_1, \ldots, f_k) \preceq c$ **do**
19 $c' \leftarrow (c_1, \ldots, c_{i-1}, f_i, \ldots, c_k)$; $C' \leftarrow C' \cup c'$; Remove c from C;
20 $C \leftarrow C \cup C'$;
21   return C;

Table 1: Datasets and Parameters used in our experiments

Dataset	Vertices	Edges	d_{1m}	d_{2m}	d_{1a}	d_{2a}
Crime	1K	1K	25	18	1.8	2.7
arXiv	39K	59K	116	18	3.5	2.6
DBpedia	225K	294K	28	12K	1.7	5.5
BookCrossing	446K	1.1M	14K	2.5K	10.9	3.4
IMDB	872K	2.7M	654	1.3K	3.9	14.6
TV Tropes	152K	3.2M	6.5K	12K	50.2	36.9
MovieLens	222K	25.0M	32.2K	81.4K	154.3	423.7

Para.	Tested values
d	1, 2, 3, 4
$\alpha(\beta)$	1, 2, 3, 4
σ	20, 40, 60, 80, 100%

Table 2: Time Complexity of ESC Algorithm

Dim	Peeling	Expanding
$d = 1$	$O(m(n + m))$	$O(m \cdot \log m + m(m + n))$
$d = 2$	$O(\epsilon \cdot (m - m/n)(n + m))$	$O(m \cdot \log m + m^2(m + n))$
$d = 3$	$O(m \cdot (m - m/n)(n + m))$	$O(\epsilon \cdot m^2(m + n) + \epsilon^2 + \epsilon \cdot m(m + n))$
$d > 3$	$O(m^{d-1} \cdot (m - m/n)(n + m))$	$O(\epsilon^{d-2} \cdot m^2(m + n) + \epsilon^{d-1} + \epsilon^{d-2} \cdot m(m + n))$

Additionally, we demonstrate the algorithm's effectiveness through a case study. All experiments are conducted on an Ubuntu server with a 2.40GHz Intel(R) Xeon(R) Gold 6240R CPU and 512GB memory in Python.

Datasets. We collected seven datasets, i.e., DBpedia, BookCrossing, IMDB, TV Tropes, arXiv, Crime, and MovieLens from the KONECT website[1]. These datasets are all undirected bipartite graphs, and neither edges nor vertices have associated significance.

Table 1 provides the statistics of datasets, of which d_{1m}, d_{2m}, d_{1a} and d_{2a} denote the maximal degree of the upper vertices, the maximal degree of the lower vertices, the average degree of the upper vertices and the average degree of the lower vertices, respectively.

As the currently available datasets lack edge weight attributes, we generated non-negative numerical multidimensional attributes for all attribute-less bipartite graph datasets, ensuring these attributes were independent. We used a uniform distribution for this purpose because it generates values with equal probability and avoids the correlation biases inherent in the Gaussian distribution. In real-world scenarios, edges with high significance in one dimension rarely exhibit high significance in others, making *anti-correlation* more common and justifying the use of uniform distribution.

Parameters. We vary three parameters: dimension (d), structural parameter (α), and dataset completeness (σ). Since α and β similarly constrain the community structure, we vary only α. Table 1 lists the parameter ranges and default values. For each experiment, we randomly selected 20 query vertex sets
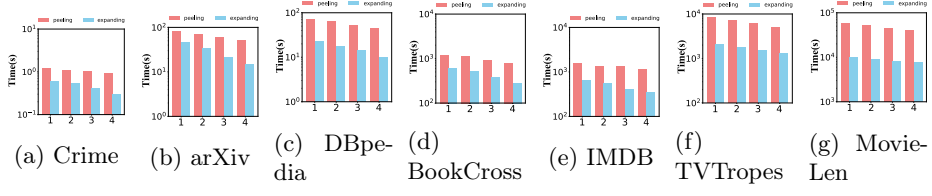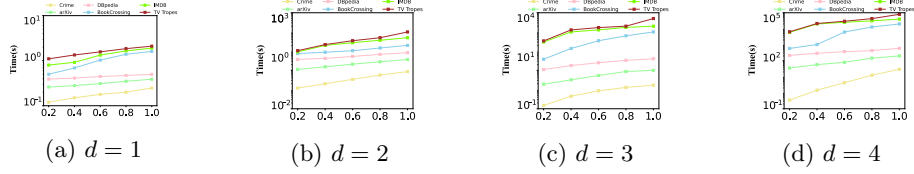
---

[1] https://konect.cc

Fig. 6: The effectiveness of the algorithms w.r.t α changes.

Fig. 7: Scalability test w.r.t varing dimensions.

from each graph that ensured the existence of the maximal (α, β)-core, and averaged the time required for these sets.

5.1 Efficiency evaluation

We provide the theoretical values of the algorithm complexity, see Table 2. We investigate the time required to search for ESCs across various datasets by varying the attribute dimensionality d while keeping other variables constant. As shown in Fig 5, we fix the structural constraint parameters α and β at 2 and use the average time taken for 20 query vertices as the search time for each dimensionality. Fig 5(a)-(g) indicates that as attribute dimensionality increases, the time required for the peeling and expanding algorithms also increases.

5.2 Effectiveness evaluation

Scalability evaluation. Without loss of generality, we take the expanding algorithm as the experimental object. To assess the scalability of Algorithm peeling $(d = 3)$, we randomly select 20%, 40%, 60%, 80%, and 100% of the edges from the input dataset(vary σ) to form subgraphs for our experiments. The experimental results confirm that the algorithm consistently produces correct outcomes. As shown in Fig 7, the algorithm's runtime generally rises with an increasing number of edges. However, deviations occur due to query vertex selection. For example, if a 20% edge subgraph fully covers a query vertex's ESC, adding more edges doesn't raise the processing time for that vertex.

Evaluating the effect of α. We maintain other variables constant and vary the value of α across different datasets. As depicted in Fig 6(a)-(g), both the peeling and expanding algorithms exhibit a decreasing trend in search time as

Edge	e_{11}	e_{12}	e_{14}	e_{15}	e_{21}	e_{22}	e_{24}	e_{26}	e_{31}	e_{33}	e_{35}	e_{36}	e_{43}	e_{44}	e_{45}	e_{46}	e_{51}	e_{52}	e_{55}	e_{56}	e_{61}	e_{63}	e_{64}	e_{66}
Attribute d=1	10	8	3	1	9	8	5	10	10	8	3	4	6	4	5	3	3	8	8	11	5	7	6	10
d=2	13	10	4	3	11	10	4	11	14	15	4	3	17	12	2	6	3	10	16	11	4	6	9	18
d=3	12	9	4	5	10	9	4	13	8	9	1	3	10	9	6	1	4	16	17	9	5	2	1	8

(a) initial state. (b) (α, β)-core.
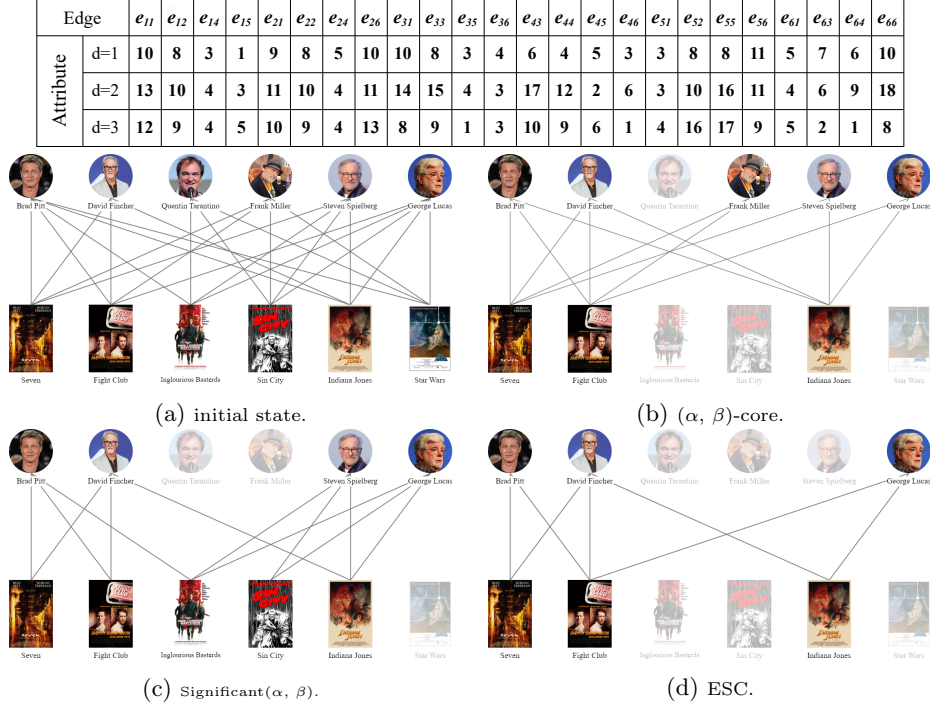
(c) Significant(α, β). (d) ESC.

Fig. 8: Case study: comparison with different models.

α increases. This trend is due to the fact that as α increases, the size of the initial maximal (α,β)-core decreases, which in turn reduces the time required by the algorithms. Additionally, we observe that the efficiency of the expanding algorithm gradually surpasses that of the peeling algorithm. This is because a higher α leads to a larger discrepancy between the size of the (α,β)-core and the original graph, resulting in the expanding algorithm requiring less time.

5.3   Case study

Comparison with different models. As shown in Figure 8, we carried out a case study on a subset of the IMDB dataset. Given that each edge in the dataset lacks attribute information, "e_{ij}" represents the edge between the i-th movie and the j-th director, we assigned three-dimensional attribute values to each edge. The first dimension signifies the duration of a director's performance, the second dimension denotes the director's remuneration, and the third dimension represents the director's rating. All three dimensions encompass synthetic attribute information. Figure 8(a) showcases the initial state of the community. When we focus on the performance of a single dimension (where $d=1$) and query vertex for *Brad Pitt* with parameters $\alpha = 2$ and $\beta = 4$, the communities identified by the algorithm proposed by [22] are illustrated in Figure 8(b). In contrast,

the communities identified by the algorithm proposed by [30] for the first dimension alone are shown in Figure 8(c). The ESCs found by our algorithm are displayed in Figure 8(d). The case study's outcomes clearly indicate that our algorithm's ESCs have a denser structure compared to those identified by [22]. Our algorithm effectively filters out edges with minimal attribute values in the second dimension, such as e_{35} and e_{36}, which highlights its precision in identifying more relevant solutions. Moreover, while [30] identifies an ESC with $f(H_1)$, our algorithm not only recognizes H_1 but also uncovers an additional ESC with $f(H_2) = (8, 10, 9)$. They are non-dominated, each excels in one dimension, underscoring that our design can discover more precise solutions in bipartite graphs with multi-dimensional attribute values than prior art[30].

6   Related Work

Community search (CS). The objective of CS is to identify cohesive subgraphs related to queries [7], which can be divided into precise algorithms and heuristic algorithms. Precise aim to find communities that meet precise optimization criteria, often involving complex mathematical modeling and calculations. Typically, models such as [3] and [38] are employed to discover precise internally cohesive communities. Heuristic use approximate methods to quickly find better communities. Some representative CS approaches are based on local information. For instance, in approach [7], the minimum degree of the k-core is utilized as a constraint for community discovery [2, 4]. Additionally, [21] takes vertex attributes into account by utilizing importance values in the CS process.

Community search on attributed graphs. Attributed graphs, categorized into k-core [20, **?**], k-truss [17, **?**], and k-clique [19] structures, are crucial for CS. [11] addressed CS on keyword attribute graphs by defining attribute communities based on structural cohesion and keyword aggregation, introducing the CL-tree index and algorithms. [10] developed a maintenance algorithm for the CL-tree index. For location graphs, [9] introduced spatial-aware communities and provided an exact solution. [12] proposed three fast algorithms for continuous spatial-aware CS on dynamic spatial graphs. In influence attribute graphs, [21] proposed an online search algorithm for the top-j influential communities with a linear space index. [4, **?**,**?**] introduced algorithms for large graphs and small j values. [13] proposed multi-attributed joint community search in road social networks, while [23] introduced the HSC model based on meta-path cohesiveness in heterogeneous graphs.

Community search on bipartite graphs. Applications of CS in bipartite graphs are largely confined to one-dimensional attributes. [30] addressed the CS problem in edge-weighted bipartite graphs by introducing the significant (α, β)-community concept, which captures vertex cohesion and maximizes the minimum edge weight within a subgraph. They proposed an indexing structure with $O(\delta \cdot m)$ for community retrieval, along with algorithms for expanding and peeling from queries to identify significant (α, β)-communities. [40] explored the CS problem in influence bipartite graphs, introducing the Pareto-optimal (α, β)-

community model, which considers both subgraph cohesion and vertex importance. It yields Pareto-optimal (α, β)-communities with vertex degree constraints and integrates Pareto optimality with $O(p \cdot m)$.

7   Conclusion

In this paper, we address the challenge of finding edge-attributed skyline communities in large bipartite graphs, which involves integrating structural constraints with attribute data to identify the most relevant subgraphs connected to a query vertex. We introduce a new model, the edge-attributed skyline community (ESC), and present an efficient peeling algorithm that iteratively removes edges with the least significant attributes across dimensions. We also propose a more effective expanding algorithm to reduce the search space and expedite the filtering of unpromising vertices, featuring a novel pseudo upper bound. Our experimental results on real-world large bipartite graphs confirm the effectiveness and efficiency of our approach in locating ESCs.

References

1. Bai, Z., Wang, M., Guo, F., Guo, Y., Cai, C., Bie, R., Jia, X.: Secmdp: Towards privacy-preserving multimodal deep learning in end-edge-cloud. In: ICDE. pp. 1659–1670. IEEE (2024)
2. Bi, F., Chang, L., Lin, X., Zhang, W.: An optimal and progressive approach to online search of top-k influential communities. arXiv:1711.05857 (2017)
3. Bonchi, F., Khan, A., Severini, L.: Distance-generalized core decomposition. In: SIGMOD. pp. 1006–1023 (2019)
4. Chen, S., Wei, R., Popova, D., Thomo, A.: Efficient computation of importance based communities in web-scale networks using a single machine. In: CIKM. pp. 1553–1562 (2016)
5. Chen, X., Wang, K., Lin, X., Zhang, W., Qin, L., Zhang, Y.: Efficiently answering reachability and path queries on temporal bipartite graphs. PVLDB **14**(10), 1845–1858 (2021)
6. Chen, Z., Zhao, Y., Yuan, L., Lin, X., Wang, K.: Index-based biclique percolation communities search on bipartite graphs. In: ICDE. pp. 2699–2712. IEEE (2023)
7. Cui, W., Xiao, Y., Wang, H., Wang, W.: Local search of communities in large graphs. In: SIGMOD. pp. 991–1002 (2014)
8. Ding, D., Li, H., Huang, Z., Mamoulis, N.: Efficient fault-tolerant group recommendation using alpha-beta-core. In: CIKM. pp. 2047–2050 (2017)
9. Fang, Y., Cheng, C., Luo, S., Hu, J., Li, X.: Effective community search over large spatial graphs. PVLDB (2017)
10. Fang, Y., Cheng, R., Li, X., Luo, S., Hu, J.: Effective community search over large spatial graphs. PVLDB **10**(6), 709–720 (2017)
11. Fang, Y., Cheng, R., Luo, S., Hu, J.: Effective community search for large attributed graphs. PVLDB **9**(12), 1233–1244 (2016)
12. Fang, Y., Wang, Z., Cheng, R., Li, X., Luo, S., Hu, J., Chen, X.: On spatial-aware community search. IEEE TKDE **31**(4), 783–798 (2018)

13. Guo, F., Yuan, Y., Wang, G., Zhao, X., Sun, H.: Multi-attributed community search in road-social networks. In: ICDE. pp. 109–120. IEEE (2021)
14. Guo, Y., Xi, Y., Wang, H., Wang, M., Wang, C., Jia, X.: Fededb: Building a federated and encrypted data store via consortium blockchains. IEEE TKDE **36**(11), 6210–6224 (2023)
15. Guo, Y., Xi, Y., Zhang, Y., Wang, M., Wang, S., Jia, X.: Towards efficient and reliable private set computation in decentralized storage. IEEE TSC (2024)
16. Guo, Y., Zhao, Y., Hou, S., Wang, C., Jia, X.: Verifying in the dark: Verifiable machine unlearning by using invisible backdoor triggers. IEEE TIFS **19**, 708–721 (2023)
17. Huang, X., Lakshmanan, L.V.: Attribute-driven community search. PVLDB **10**(9), 949–960 (2017)
18. Ley, M.: The dblp computer science bibliography: Evolution, research issues, perspectives. In: International symposium on string processing and information retrieval. pp. 1–10 (2002)
19. Li, J., Wang, X., Deng, K., Yang, X., Yu, J.X.: Most influential community search over large social networks. In: ICDE (2017)
20. Li, R., Qin, L., Ye, F., Yu, J.X., Xiao, X., Xiao, N., Zheng, Z.: Skyline community search in multi-valued networks. In: SIGMOD. pp. 457–472 (2018)
21. Li, R., Qin, L., Yu, J.X., Mao, R.: Influential community search in large networks. PVLDB **8**(5), 509–520 (2015)
22. Liu, Q., Liao, X., Huang, X., Xu, J., Gao, Y.: Distributed (α, β)-core decomposition over bipartite graphs. In: IEEE ICDE. pp. 909–921. IEEE (2023)
23. Liu, Y., Guo, F., Xu, B., Bao, P., Shen, H., Cheng, X.: Sach: Significant-attributed community search in heterogeneous information networks. In: ICDE. pp. 3283–3296. IEEE (2024)
24. Luo, J., Cao, X., Xie, X., Qu, Q., Xu, Z., Jensen, C.S.: Efficient attribute-constrained co-located community search. In: ICDE. pp. 1201–1212 (2020)
25. Meng, L., Shao, Y., Yuan, L., Lai, L., Cheng, P., Li, X., Yu, W., Zhang, W., Lin, X., Zhou, J.: A survey of distributed graph algorithms on massive graphs. ACM Computing Surveys **57**(2), 1–39 (2024)
26. Sarıyüce, A.E., Pinar, A.: Peeling bipartite networks for dense subgraph discovery. In: WSDM. pp. 504–512 (2018)
27. Wang, J., De Vries, A.P., Reinders, M.J.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: SIGIR. pp. 501–508 (2006)
28. Wang, K., Lin, X., Qin, L., Zhang, W., Zhang, Y.: Efficient bitruss decomposition for large-scale bipartite graphs. In: ICDE. pp. 661–672 (2020)
29. Wang, K., Zhang, W., Lin, X., Qin, L., Zhou, A.: Efficient personalized maximum biclique search. In: ICDE. pp. 498–511 (2022)
30. Wang, K., Zhang, W., Lin, X., Zhang, Y., Qin, L., Zhang, Y.: Efficient and effective community search on large-scale bipartite graphs. In: ICDE. pp. 85–96. IEEE (2021)
31. Wang, K., Zhang, W., Zhang, Y., Qin, L., Zhang, Y.: Discovering significant communities on bipartite graphs: An index-based approach. IEEE TKDE **35**(3), 2471–2485 (2023)
32. Xu, G., Kong, D.l., Zhang, K., Xu, S., Cao, Y., Mao, Y., Duan, J., Kang, J., Chen, X.B.: A model value transfer incentive mechanism for federated learning with smart contracts in aiot. IEEE IoTJ (2024)
33. Xu, S., Cao, Y., Chen, X., Guo, Y., Yang, Y., Guo, F., Yiu, S.M.: Post-quantum searchable encryption supporting user-authorization for outsourced data management. In: ACM CIKM. pp. 2702–2711 (2024)

34. Xu, S., Chen, X., Guo, Y., Yang, Y., Wang, S., Yiu, S.M., Cheng, X.: Lattice-based forward secure multi-user authenticated searchable encryption for cloud storage systems. IEEE Transactions on Computers (2025)
35. Xu, T., Xu, S., Chen, X., Chen, F., Li, H.: Multi-core token mixer: a novel approach for underwater image enhancement. Machine Vision and Applications **36**(2), 1–16 (2025)
36. Xu, Z., Zhang, Y., Yuan, L., Qian, Y., Chen, Z., Zhou, M., Mao, Q., Pan, W.: Effective community search on large attributed bipartite graphs. International Journal of Pattern Recognition and Artificial Intelligence **37**(02), 2359002 (2023)
37. Yuan, L., Xu, J., Chen, Z., Ma, C., Xu, J., Qin, L.: Efficient maximum balanced k-biplex search over bipartite graphs. In: ICDE. pp. 3098–3112. IEEE (2025)
38. Zhang, Y., Yu, J.X.: Unboundedness and efficiency of truss maintenance in evolving graphs. In: SIGMOD. pp. 1024–1041 (2019)
39. Zhang, Y., Phillips, C.A., Rogers, G.L., Baker, E.J., Chesler, E.J., Langston, M.A.: On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types. BMC bioinformatics **15**, 1–18 (2014)
40. Zhang, Y., Wang, K., Zhang, W., Lin, X., Zhang, Y.: Pareto-optimal community search on large bipartite graphs. In: CIKM. pp. 2647–2656 (2021)
41. Zhou, Y., Fang, Y., Luo, W., Ye, Y.: Influential community search over large heterogeneous information networks. PVLDB **16**(8), 2047–2060 (2023)