The Knowledge Engineering Review

RESEARCH ARTICLE

Open Access

https://doi.org/10.48130/ker-0026-0008

The Knowledge Engineering Review **2026**, 41: e010

Top-*r* semantically important community search on semantic-rich heterogeneous graphs

Dandan Zhang[1], Yuxiang Wang[1*], Chengjie Gu[1], Xiangyu Ke[2], Xiaoliang Xu[1] and Qiming Fang[1]

[1] *School of Computer Science (School of Software), Hangzhou Dianzi University, Hangzhou 310018, China*
[2] *School of Software Technology, Zhejiang University, Hangzhou 310027, China*
* Correspondence: lsswyx@hdu.edu.cn (Wang Y)

Abstract

Given a heterogeneous information network (HIN) G and a query node q, community search (CS) on HINs relies on a predefined symmetric meta-path $\mathcal{P}$ to identify a community from G that contains q, where all nodes are connected via instances of $\mathcal{P}$. In semantic-rich HINs, structurally different meta-paths can convey similar semantics. Relying solely on a single meta-path may therefore cause the loss of similar relational semantics in the result community and fail to cover all possible community members. Worse still, existing methods only return one community that strictly adheres to a given $\mathcal{P}$, leading to limited semantic diversity in the results. This inspires us to study the top-r semantically important community search (r-SICS) problem, based on a generic and flexible semantic meta-path pattern (SMP), which allows users to choose from multiple meaningful communities with diverse relational semantics. We first propose the `Basic` algorithm, which gradually finds top-r semantically important communities (SICs) from a small q-centric SMP-graph G_{SMP} instead of the entire G, with the semantic relevance to SMP from large to small. Then, we optimize `Basic` with a semantic-level binary search strategy to accelerate G_{SMP} construction and an intermediate recording strategy to reduce repeated SIC construction from G_{SMP} in binary semantic search. Besides, we present a parallelization strategy to further enhance efficiency. Extensive experimental studies on four real-world, million-scale datasets validated the effectiveness and efficiency of our methods.

Keywords: Top-*r* community search, Semantic-rich HINs, Semantic meta-path pattern, Semantically important community

Citation: Zhang D, Wang Y, Gu C, Ke X, Xu X, et al. 2026. Top-*r* semantically important community search on semantic-rich heterogeneous graphs. *The Knowledge Engineering Review* 41: e010 https://doi.org/10.48130/ker-0026-0008

1 Introduction

Heterogeneous information networks (HINs) are widely used to represent complex real-world entities (nodes) and their relations (edges) in various domains[1–5], such as social networks[6,7], bibliographic networks[8,9], and knowledge graphs[10,11]. Community search (CS) on HINs is a fundamental problem in graph mining[12–14], which identifies a cohesive subgraph from a given large-scale HIN G that contains the user-specified query node q with high structural cohesiveness and semantic quality[13,15,16]. CS on HINs has broad applicability across multiple domains, such as marketing advertisement and event planning[1,17], expert finding[16,18], biological analysis[19,20], and GraphRAG for retrieval-augmented LLM[21–23].

Motivations. Existing methods for CS on HINs primarily rely on predefined symmetric meta-paths to identify communities adhering to specific relational semantics[1,3,15,16,18,24,25]. A symmetric meta-path $\mathcal{P}$ is a sequence of node- and edge-types satisfying that $\mathcal{P}$ is the same as its reverse, i.e., $\mathcal{P}^{-1} = \mathcal{P}$[1]. In Fig. 1, v_1 (Rihanna) and v_5 (Eminem) are linked via a 2-hop path instance $p = v_1v_6v_5$ of a symmetric $\mathcal{P} = \text{Person}^{\text{Artist}}\text{Song}^{\text{Artist}}\text{Person}$. By integrating symmetric meta-paths with classic community models, such as k-core[26,27] or k-truss[28,29], a series of meta-path-based variants like $(k,\mathcal{P})$-core[1,18] and $(k,\mathcal{P})$-truss[25] have emerged. A $(k,\mathcal{P})$-core is a subgraph where each node has at least k $\mathcal{P}$-neighbors, and a $(k,\mathcal{P})$-truss is a subgraph where any two adjacent nodes share at least $k-2$ $\mathcal{P}$-neighbors. Two nodes are $\mathcal{P}$-neighbors if they are connected via a path instance of $\mathcal{P}$, e.g., v_1 and v_5 are $\mathcal{P}$-neighbors given the above symmetric meta-path. One can easily come up with a CS query with $q = v_1$, $\mathcal{P} = \text{Person}^{\text{Artist}}\text{Song}^{\text{Artist}}\text{Person}$, and $k = 2$. The subgraph that consists of $\{v_1, \cdots, v_6\}$ would be identified as a connected $(2,\mathcal{P})$-core community, where every Person-typed node has 2 $\mathcal{P}$-neighbors. Although easy to implement, it suffers from two issues that undermine its usability, as shown in the following two examples in practice.

Loss of similar relational semantics. In semantic-rich HINs like DBpedia, which contain hundreds (or even more) of node- and edge-types, and forms over thousands of frequently used (both symmetric and asymmetric) meta-paths. It is common for a single relationship to be represented by multiple structurally different but semantically similar meta-paths[30]. In such cases, existing methods that rely solely on a single symmetric meta-path would lose similar relational semantics in community results, thereby failing to cover all possible community members (see Example 1).

Example 1 *Figure 1 illustrates a snapshot of DBpedia in the entertainment industry, with various semantically similar meta-paths that capture collaborative relations between musicians. Representative examples include* $\mathcal{P}_1 = \text{Person}^{\text{Artist}}\text{Song}^{\text{Artist}}\text{Person}$, $\mathcal{P}_2 = \text{Person}^{\text{Artist}}\text{Song}^{\text{Writer}}\text{Person}$, $\mathcal{P}_3 = \text{Person}^{\text{Artist}}\text{Song}^{\text{Composer}}\text{Person}$, *etc. Given a CS query with* $q = v_1$, $\mathcal{P} = \mathcal{P}_1$, *and* $k = 2$, *only a relatively small community of musicians* $\{v_1, v_3, v_5\}$ *is returned. Notably excluded are several musicians with known close collaborations to* Rihanna, *such as* The Dream *and* JAY – Z. *This occurs because the meta-paths connecting these nodes involve asymmetric* $\mathcal{P}_3$, *while the CS query is restricted to* $\mathcal{P} = \mathcal{P}_1$, *causing these nodes to have fewer than* k $\mathcal{P}$*-neighbors and thus be consequently omitted from the result.*

A straightforward solution is to enumerate all semantically similar, symmetric and asymmetric meta-paths from thousands of candidate meta-paths for running CS on semantic-rich HINs, then combine the results, which is evidently time-consuming and impractical for both ordinary and expert users. A more efficient solution is to implement a lightweight semantic representation capable of flexibly matching

https://doi.org/10.48130/ker-0026-0008

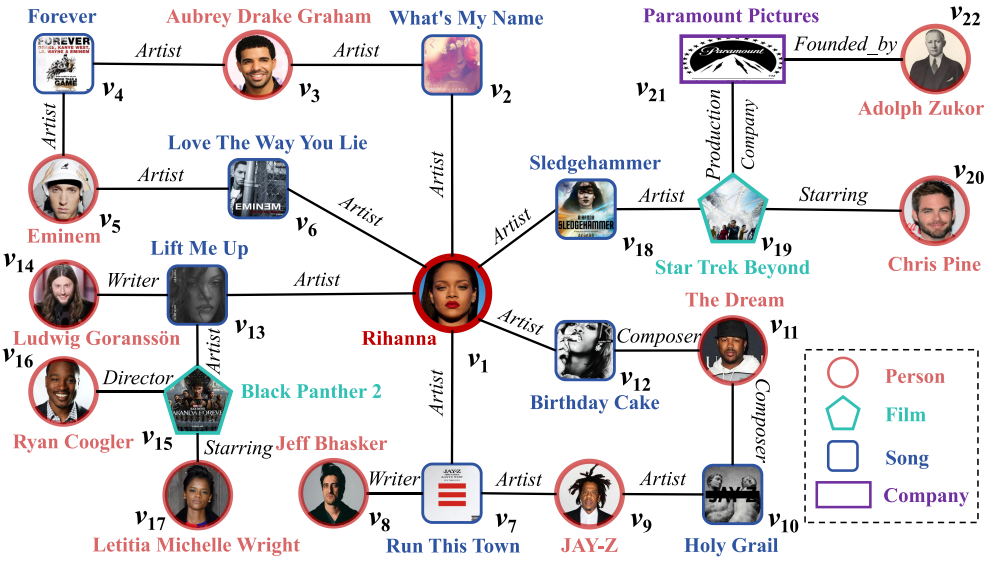

Fig. 1 An example of the semantic-rich HIN DBpedia.

various semantically similar meta-paths in runtime. *This motivates us to employ a more generalized Semantic Meta-path Pattern (SMP), which abstracts a set of symmetric and asymmetric meta-paths using only key elements in representing their semantics without predefining precise meta-paths in advance.*

Homogeneity of the community results. In practice, users often have their own specific preferences on the inherent semantics and structure characteristics of the returned community. Some may prefer a compact community with high semantic relevance to the given $\mathcal{P}$, while others may prefer a larger community with diverse relational semantics among its members. However, existing works only return one community strictly adhering to the given $\mathcal{P}$, which cannot provide a variety of options for selection.

Example 2 *Figure 2 (middle) shows a homogeneous graph of the HIN in Fig. 1, where each colored edge between* Person*-typed nodes is a path instance of a meta-path in Fig. 2 (left). Given* $q = v_1$*, some users may prefer a pure singer community of* Rihanna *involving* $\{v_1, v_3, v_5\}$ *(H_1), where nodes are exclusively connected via path instances of* $\mathcal{P}_1$*. Some users may prefer a more general community, such as* H_2 *and* H_3*, which focuses on musical collaboration in a broad sense including songwriters and composers, respectively. Furthermore, some users might even be interested in the broader pan-entertainment community of* Rihanna*, which includes film directors and actors. In that case,* H_4 *and* H_5 *would be their desired communities.*

Compared with returning only a single community, offering the best r (≥ 1) communities with broad semantic diversity that are most semantically relevant to user's query intention is a more effective and practical solution. *This motivates us to study the top-r semantically important community search (r-SICS) problem based on the proposed SMP, where the importance of a community is directly reflected by its semantic relevance to the query in real applications.*

Applications. r-SICS can discover high-quality communities with semantic diversity in many applications. (1) *Research group detection*[5,31]. In academic peer review, r-SICS can identify r research groups of varying collaborative closeness associated with an author, which helps minimize potential conflicts of interest in reviewer assignment. (2) *Criminal network disruption*[32–34]. In the field of public security, r-SICS can pinpoint r organizations of varying closeness with respect to specific social relationships to which a suspect belongs. This provides rich and prioritized investigative leads, enhancing the effectiveness of criminal investigation. (3) *Community-based RAG*[22,23]. For LLM-driven question answering, r-SICS can retrieve r knowledge communities with varying semantic relevance to the query, which helps achieve a better balance between the quality and diversity of generated answers.

Challenges and contributions. Existing community search methods on heterogeneous information networks typically rely on a predefined meta-path $\mathcal{P}$ and aim to identify a $(k,\mathcal{P})$-core structure to capture a given semantic pattern, implicitly assuming that the semantic scope

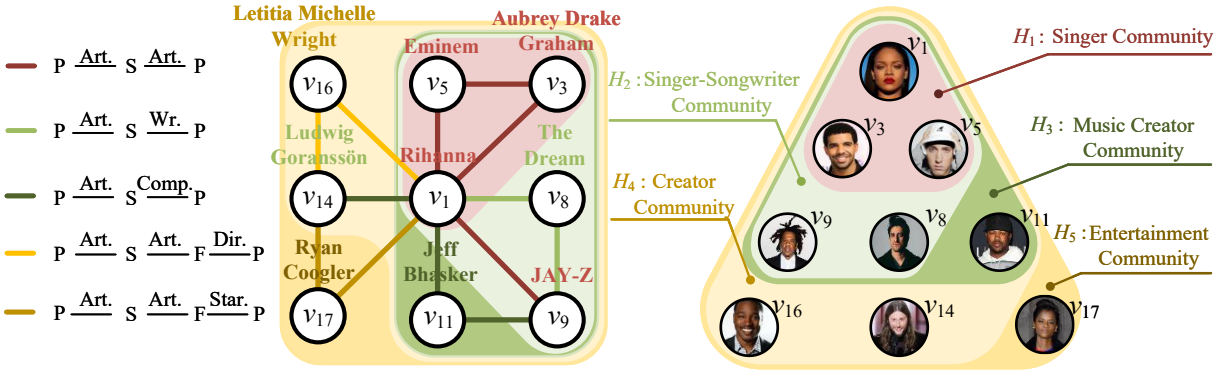

Fig. 2 Diverse semantically similar communities.

is explicitly specified at query time by the user-input $\mathcal{P}$. In our setting, however, the user is only required to provide a general Semantic Meta-path Pattern (SMP) as an abstract description of their semantic expectation. Rather than conducting a deterministic search over a fixed set of meta-paths, the query process is guided by this semantic expectation to progressively explore, refine, and dynamically discover potential semantic relations. As a result, traditional community search methods, which rely on explicitly defined $\mathcal{P}$ at query time, are no longer applicable. Motivated by this observation, we adopt a different paradigm in which community construction is tightly coupled with the exploration of potential semantic relations. Specifically, we propose a simple baseline method `Basic` (section 3), which constructs a compact, q-centric semantic meta-path subgraph G_{SMP} by retaining only those meta-path instances that match the user-specified SMP. As semantic relations are continuously explored and progressively refined, G_{SMP} is dynamically expanded, and semantically important communities are repeatedly mined within this restricted and evolving subgraph. In doing so, `Basic` avoids the exponential cost of enumerating all similar semantic subgraphs, while enabling efficient discovery of candidate semantic relations and community structures. The smaller the G_{SMP}, the faster the `Basic`. To further improve the efficiency of G_{SMP} construction, we present an optimized algorithm `Opt-BSS` with binary semantic search (section 4.1). It constructs G_{SMP} by processing edge-types in batches and applies a semantic-level binary search to iteratively extract a smaller, inherent G_{SMP} from the maximal one, avoiding repeated G_{SMP} construction from scratch. We next extend `Opt-BSS` with a recording optimization that reuses intermediate information during semantic-level binary search, called `Opt-Record` (section 4.2), avoiding the SIC construction from G_{SMP}. Finally, to improve practical efficiency, we further incorporate a multithreaded parallel expansion strategy for G_{SMP} construction as a system-level acceleration technique. This implementation-oriented efficiency enhancement is designed to improve scalability in large-scale graphs and complements the main algorithmic framework.

Overall, we conclude our contributions as follows.

- We are the first to study the r-SICS problem based on a generic and flexible semantic meta-path pattern (SMP) to return top-r communities with broad semantic diversity in a unified and efficient manner.
- We propose the `Basic` algorithm, which gradually finds top-r communities from a q-centric SMP-graph G_{SMP}, with the semantic relevance to SMP from large to small.
- We optimize `Basic` algorithm with a semantic-level binary search strategy (`Opt-BSS`) to improve the efficiency of G_{SMP} construction and intermediate information recording strategy (`Opt-Record`) to reduce repeated SIC construction from G_{SMP} in binary semantic search.
- As an implementation-level efficiency enhancement, we further introduce a parallel expansion strategy to accelerate G_{SMP} construction in large-scale graphs.
- We conducted extensive experiments on four real-world, million-scale datasets and validated the effectiveness and efficiency of the proposed methods.

2 Preliminaries and problem

Definition 1 (Heterogeneous Information Network [HIN][3,16]). *An HIN is defined as $G = (V, E)$, where V (E) is the node (edge) set. Each node $u \in V$ has a type $\phi(v)$ through a mapping function $\phi : V \to \mathcal{A}$, where $\mathcal{A}$ denotes all node types. And each edge $e \in E$ has a type $\psi(e)$ through a mapping function $\psi : E \to \mathcal{R}$, where $\mathcal{R}$ denotes all edge types. An HIN satisfies that $|\mathcal{A}| > 1$ and $|\mathcal{R}| > 1$.*

Definition 2 (HIN Schema[3,16]). *Given an HIN $G = (V, E)$ with mapping functions $\phi : V \to \mathcal{A}$ and $\psi : E \to \mathcal{R}$, its schema $T_G = (\mathcal{A}, \mathcal{R})$ is defined as a graph over $\mathcal{A}$ and $\mathcal{R}$.*

Example 3 *Figure 3 (left) shows the schema of the HIN in Fig. 1. It includes four node types $\mathcal{A}$: Person (P), Film (F), Song (S), Company (C). For example, v_1 stands for Rihanna with $\phi(v_1) = P$, as well as seven edge types $\mathcal{R}$: Artist (Art.), Writer (Writ.), Composer (Comp.), Director (Dir.), Starring (Star.), Production Company (Prod.Co) and Founded_by (Found.By), e.g., $\psi(e_{v_1 v_2})$ = Art., indicating that Rihanna (v_1) is the artist of What's My Name (v_2).*

In HINs, communities are widely prevalent cohesive subgraphs, whose cohesiveness is evaluated by **semantic cohesiveness** and **structural cohesiveness**, as discussed.

2.1 Semantic cohesiveness

We start with the meta-path, which is commonly used to represent the semantic relationships between two node types.

Definition 3 (Meta-path[1,18]). *A meta-path $\mathcal{P}$ is a path defined on an HIN schema $T_G = (\mathcal{A}, \mathcal{R})$, and is denoted in the form $A_1 \underline{R_1} A_2 \underline{R_2} \cdots \underline{R_l} A_{l+1}$, where l is the length of $\mathcal{P}$, $A_i \in \mathcal{A}$ $(1 \le i \le l+1)$ and $\underline{R_j} \in \mathcal{R}$ $(1 \le j \le l)$.*

We denote a path between v_1 and v_l as $p_{v_1 v_l} = v_1 \cdots v_l$. If $p_{v_1 v_l}$ conforms to a $\mathcal{P}$, it is called a path instance of $\mathcal{P}$.

In semantic-rich HINs, various meta-paths can describe the same type of relationship. For instance, the four meta-paths in Fig. 3 (right) all represent collaboration in the entertainment industry, yet reflect distinct semantics due to: (1) Domain differences: $\mathcal{P}_1 \sim \mathcal{P}_3$ primarily focus on the music domain, while $\mathcal{P}_4$ pertains to the film. (2) Even within the same domain, differences in roles can lead to variations in collaborative semantics. e.g., $\mathcal{P}_1$ indicates co-artist collaboration, $\mathcal{P}_2$ and $\mathcal{P}_3$ involve artist-songwriter/composer collaborations. Nevertheless, all of them consistently express the collaborative semantics in music creation. To precisely capture the user's desired relational semantics for a community of interest, it is necessary to enumerate as many semantically similar meta-paths as possible. However, this is extremely challenging in semantic-rich HINs. For instance, in

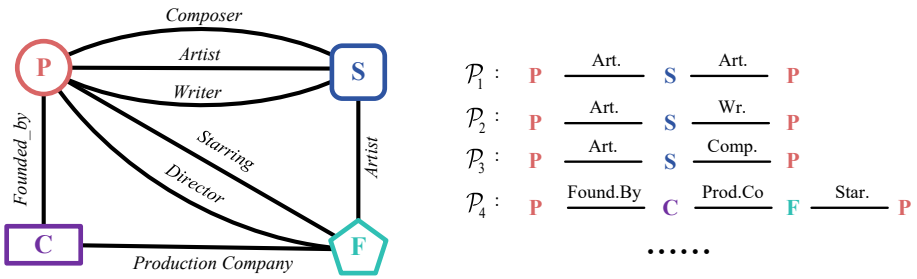

Fig. 3 The schema of the HIN illustrated in Fig. 1 with five example meta-paths between nodes with a type `Person`.

https://doi.org/10.48130/ker-0026-0008

DBpedia (with 359 node types and 676 edge types), it has over 1,000 frequently used meta-paths, with the total number of meta-paths within 4 hops exceeding 50,000. Accurately predefining meta-paths is impractical in semantic-rich HINs.

Therefore, we employ a more generalized **Semantic Meta-path Pattern** (SMP) instead of relying on specific meta-paths, to extract key elements that characterize the semantics of meta-paths and better capture the intended semantics of user interests.

Definition 4 (Semantic Meta-path Pattern [SMP]). *An SMP is defined as a quadruple* $\langle A_t, A_a, S, \hat{l} \rangle$ *is the end node type of a semantic meta-path, which represents the target members of a community of interest. (1)* $A_t \in \mathcal{A}$ *is the end node type of a meta-path, representing the target members of a community. (2)* $A_a \in \mathcal{A}$ *is defined as the anchor node type located in the middle of the semantic meta-path, serving to specify its domain. (3)* $S \in \mathcal{R}$ *is an edge type indicating the semantics in which the user is interested. (4)* $\hat{l}$ *is the upper bound of the length of a meta-path. It is designed to prevent excessive meta-path length.*

A meta-path structurally matches an SMP if it starts and ends with A_t, passes through A_a, and has a length within $\hat{l}$ hops, denoted as $\mathcal{P} \models$ SMP; otherwise, $\mathcal{P} \not\models$ SMP. The SMP imposes only a coarse-grained semantic constraint on candidate meta-paths; its role is to capture an abstract representation of the user's semantic expectation, rather than to provide a complete or precise characterization of the user's intended semantics.

Example 4 *To find a collaboration community in the music industry from DBpedia, a user may come up with a simple SMP =* $\langle$*Person, Song, Artist, 3*$\rangle$*, where* A_t *= Person,* A_a *= Song,* S *= Artist,* $\hat{l}$ *= 3. This SMP indicates that the community consists of individuals (*Person*) engaged in the music domain (*Song*) and connected via relations similar to* Artist *within a collaboration range of 3 hops. Figure 3 (right) shows three meta-paths* $\mathcal{P}_1 \sim \mathcal{P}_3$ *matching this SMP.*

Although multiple meta-paths may structurally conform to a given SMP, they can exhibit substantially different degrees of semantic relevance to the user intent. Therefore, rather than treating SMP matching as the final criterion, we introduce **Meta-path Semantic Similarity** to explicitly quantify how well a meta-path aligns with the target semantics, denoted as $W_{\mathcal{P}\to\text{SMP}}$.

Definition 5 (Meta-path Semantic Similarity). *Given an* l*-hop meta-path* $\mathcal{P}$ *that matches an SMP =* $\langle A_t, A_a, S, \hat{l} \rangle$*, its semantic similarity to the SMP, denoted by* $W_{\mathcal{P}\to\text{SMP}}$*, characterizes the actual semantic relevance of* $\mathcal{P}$ *to the user intent. We define it to be dominated by the minimum edge-type similarity to* S *(Eq. 1), where* S_i *(*$i \in [1, l]$*) denotes the* i*-th edge type in* $\mathcal{P}$*, and* $W_{S_i\to S}$ *represents the semantic similarity between* S_i *and* S*.*

$$W_{\mathcal{P}\to\text{SMP}} = \min\{W_{S_1\to S}, W_{S_2\to S}, \cdots, W_{S_l\to S}\} \tag{1}$$

This design is based on the intuition that a meta-path instance can serve as a reliable semantic connection only when all of its constituent relations remain compatible with the target semantics. Once one edge type becomes semantically weak or mismatched, the semantic interpretation of the whole path is correspondingly constrained. Thus, the minimum value is used not to oversimplify the semantics of a meta-path instance, but to provide a conservative estimate of its overall semantic validity. In addition, this minimum-based formulation is widely adopted in graph mining due to its effectiveness[4], and is computationally simpler than alternative aggregation measures such as the geometric mean[30,35], making it particularly suitable for large-scale graphs. To compute $W_{\mathcal{P}\to\text{SMP}}$, we first need to compute $W_{S_i\to S}$: (1) we employ an offline heterogeneous graph embedding model, e.g., TransE[36] and RotatE[37], to represent each S_i in an HIN G as a d-dimensional vector $\vec{S}_i$, which effectively preserves the intrinsic semantics of original relations in vectors[38]; (2) $W_{S_i\to S} \in [0,1]$ is then computed as Eq. 2.

$$W_{S_i\to S} = \frac{\vec{S}_i \cdot \vec{S}}{\|\vec{S}_i\| \, \|\vec{S}\|} \tag{2}$$

Example 5 *We provide the semantic similarity of the meta-paths shown in Fig. 3 to the SMP =* $\langle$*Person,Song,Artist,3*$\rangle$*. Given that* $W_{\text{Art.}\to\text{Art.}} = 1.0$ *and* $W_{\text{Wr.}\to\text{Art.}} = 0.95$ *(obtained via TransE embedding model), we have* $W_{\mathcal{P}_1\to\text{SMP}} = \min\{1.0, 1.0\} = 1.0$ *and* $W_{\mathcal{P}_2\to\text{SMP}} = \min\{1.0, 0.95\} = 0.95$*. This indicates that* $\mathcal{P}_1$ *is more semantically similar to SMP than* $\mathcal{P}_2$*. Besides,* $W_{\mathcal{P}_4\to\text{SMP}} = 0$ *as* $\mathcal{P}_4 \not\models$ SMP.

Semantic cohesiveness of two nodes. Given two nodes $u, v \in V$ connected by a path instance p_{uv} conforming to a meta-path $\mathcal{P}_{uv} \models$ SMP, the semantic cohesiveness between u and v with respect to the SMP is determined by its corresponding meta-path's semantic similarity to SMP, i.e., $W_{\mathcal{P}_{uv}\to\text{SMP}}$. The greater the similarity of $\mathcal{P}_{uv}$ to SMP, the cohesiveness between u and v.

In practical scenarios, multiple meta-path instances conforming to the SMP often exist between the same pair of nodes. Generally, if a path contains any semantically mismatched or weak edge type, the path can hardly be regarded as semantically valid, and its overall semantic connection can be considered 'broken'. Meanwhile, when multiple relational explanations exist between two nodes, weaker relations should not negate the existence of a stronger and more semantically consistent one; the presence of at least one highly coherent and tightly coupled semantic path is sufficient to indicate a strong semantic association between the two nodes. Based on these intuitions, we adopt the following design: when multiple meta-paths connect nodes u and v, the semantic similarity of each meta-path is determined by its weakest edge type, and the overall semantic cohesiveness between u and v is defined as the maximum semantic similarity among all matching meta-paths.

Semantic cohesiveness of a subgraph. We next formally define the semantic cohesiveness of a subgraph as follows.

Definition 6 (Semantic Cohesiveness of a Graph). *Given a subgraph* $H \subseteq G$ *and an SMP, the semantic cohesiveness of* H *is defined as the minimum semantic cohesiveness of any two nodes in* H*, i.e.,* $W_{H\to\text{SMP}} = \min_{u,v\in V} W_{\mathcal{P}_{uv}\to\text{SMP}}$.

Example 6 *Recall the graph in Fig. 2 (middle). For SMP =* $\langle$*Person,Song,Artist,3*$\rangle$*, if we consider the subgraph* H_1 *that consists of* $\{v_1, v_3, v_5\}$*, then* $W_{H_1\to\text{SMP}} = 1.0$ *as each pair of nodes exhibits the same high semantic cohesiveness with respect to SMP dominating by the meta-path* $\mathcal{P} =$ Person$\overset{\text{Artist}}{}$Song$\overset{\text{Artist}}{}$Person*. Similarly, when we consider the subgraph* H_2*, then* $W_{H_2\to\text{SMP}} = 0.95$ *as the semantic cohesiveness between* v_1 *and* v_8 *(or* v_8 *and* v_9*) is the worst due to they are connected via the path instance of* Person$\overset{\text{Artist}}{}$Song$\overset{\text{Write}}{}$Person *with* $W_{\text{Wr.}\to\text{Art.}} = 0.95$.

2.2 Structural cohesiveness

We next integrate the SMP with the widely used k-core model to measure community structure cohesiveness from a structural perspective.

Definition 7 ($\mathcal{P}^*$-neighbor). *Given an SMP and two nodes* u *and* v *in an HIN, if* u *is connected to* v *via a path instance* p_{uv} *of* $\mathcal{P}^* \models$ SMP*, then* v *is a* $\mathcal{P}^*$*-neighbor of* u*, where* $\mathcal{P}^*$ *is a wildcard representing any meta-path that matches the SMP.*

Definition 8 ($\mathcal{P}^*$-edge). *Given an SMP and two nodes* u *and* v *in an HIN that are mutual* $\mathcal{P}^*$*-neighbors, the path instance* p_{uv} *of* $\mathcal{P}^* \models$ SMP *between them is called a* $\mathcal{P}^*$*-edge.*

Definition 9 (Connected (k, SMP)-core). *Given an HIN* G*, an SMP, and an integer* $k \geq 0$*, a connected* $(k,$SMP$)$*-core* $H \subseteq G$ *is a*

connected subgraph of nodes linked by $\mathcal{P}^*$-edges, of which every node u has at least k distinct $\mathcal{P}^*$-neighbors.

Notably, k serves only as a minimum degree constraint, and the core is not required to achieve the maximum possible k.

Example 7 *Given an SMP = $\langle$Person, Song, Artist, 3$\rangle$ and $k = 2$, Fig. 2 (middle) shows a connected (2, SMP)-core extracted from the HIN in Fig. 1. Each Person-typed node has at least two $\mathcal{P}^*$-neighbors ($\mathcal{P}^*$ refers to any of the five meta-paths in Fig. 2 (left) that match the given SMP), e.g., node v_1 (Rihanna) has eight distinct $\mathcal{P}^*$-neighbors (i.e., v_3, v_5, v_8, v_9, v_{11}, v_{14}, v_{16}, v_{17}), the $\mathcal{P}^*$-edge between v_1 and v_5 (Eminem) is* Person$^{\text{Artist}}$Song$^{\text{Artist}}$Person.

The connected (k, SMP)-core generalizes the traditional (k, $\mathcal{P}$)-core by allowing any meta-path $\mathcal{P}^* \models$ SMP, rather than being restricted to a single $\mathcal{P}$. This, in a more flexible and expressive manner, enhances the semantic diversity of a community.

2.3 Problem definition

We first introduce the concept of **S**emantically **I**mportant **C**ommunity (SIC), which is defined based on the semantic cohesiveness and the connected (k,SMP)-core.

Definition 10 (Semantically Important Community [SIC]). *Given an HIN $G = (V, E)$, an SMP = $\langle A_t, A_a, S, \hat{l}\rangle$, a query node q with type A_t, and an integer $k \geq 0$, a subgraph $H \subseteq G$ is called an SIC if it satisfies the following conditions:*

- ***Query Participation:*** *The subgraph H contains q;*
- ***Structure cohesiveness:*** *H is a connected (k,SMP)-core;*
- ***Semantic cohesiveness:*** *The semantic cohesiveness of H is quantified by $W_{H\to\text{SMP}}$, where a larger $W_{H\to\text{SMP}}$ implys a stronger semantic importance of H to SMP;*
- ***Maximality:*** *There does not exist another subgraph $H' \supseteq H$ satisfying the first two conditions and having the same semantic cohesiveness as $W_{H\to\text{SMP}}$.*

More precisely, an SIC is defined as the largest one among all connected (k,SMP)-cores that share an equivalent level of semantic cohesiveness (or importance) to a specific SMP. This definition enables SIC to encompass more diverse semantics without compromising semantic importance, enhancing the semantic diversity.

Example 8 *The five subgraphs $H_1 \sim H_5$ in Fig. 2 (right) represent distinct SICs with varying semantic cohesiveness to a given SMP = $\langle$Person, Song, Artist, 3$\rangle$, as introducing additional $\mathcal{P}^*$-neighbors or $\mathcal{P}^*$-edges to any H would not yield a larger subgraph that maintains the same level of semantic cohesiveness. Besides, taking $H_2 = \{v_1, v_3, v_5, v_8, v_9\}$ as an example, it also contains a connected (2, SMP)-core of $\{v_1, v_8, v_9\}$. Although both share the same semantic cohesiveness dominated by* Person$^{\text{Artist}}$Song$^{\text{Write}}$Person, *H_2 includes more nodes connected via instances of the meta-path with higher semantic similarity to SMP, exhibiting better semantic diversity. In our setting, H_2 would be the SIC rather than the smaller one.*

In real-world scenarios, a node may belong to multiple SICs with varying semantic cohesiveness to a given SMP, reflecting hierarchical semantic scopes. For instance, in Fig. 2, v_1 (Rihanna) belongs to a pure singer community (H_1), a broader creative musician community encompassing songwriters and composers (H_2 and H_3), and even a more generalized entertainment community (H_4 and H_5). In practice, users are often uncertain about how broad the desired community semantics should be at query time. Some may prefer the most semantically focused community that best matches their initial intent, while others may further expect broader communities that include more semantically related members. This motivates us to study the Top-r **S**emantically **I**mportant **C**ommunity **S**earch (r-SICS) problem, which explores the top-r SICs most relevant to the user-specific SMP and builds a multi-level answer space for user exploration. As r increases, the returned communities become progressively broader while preserving a semantic ordering from high to low importance, thereby supporting a natural exploration process from precise semantic matches to broader semantic neighborhoods.

***r*-SICS Problem**. Given an HIN $G = (V, E)$, a semantic meta-path pattern SMP = $\langle A_t, A_a, S, \hat{l}\rangle$, a query node q with type A_t, an integer $k \geq 0$ and an integer $r \geq 1$, r-SICS aims to find the best r SICs that satisfy the following requirements:

- **Size monotonicity:** The size of SIC satisfies $|V_i| < |V_{i+1}|$ ($|V_i|$ is the number of nodes in the i-th SIC, $1 \leq i < r$);
- **Semantic monotonicity:** The semantic cohesiveness of SIC must satisfy $W_{H_i\to\text{SMP}} > W_{H_{i+1}\to\text{SMP}}$ ($W_{H_i\to\text{SMP}}$ is the semantic cohesiveness of the i-th SIC, $1 \leq i < r$).

Example 9 *Given an SMP = $\langle Person, Song, Artist, 3\rangle$, $k = 2$, $r = 1$, and a query node v_1 (Rihanna), r-SICS returns H_1 as the top-1 SIC, clearly revealing the critical pure singer community of Rihanna. For $r = 3$, H_2 and H_3 are also returned, including more songwriters and composers who have strong musical collaborations with Rihanna. For $r = 5$, broader pan-entertainment communities H_4 and H_5 of Rihanna are identified, which involve many film directors and actors.*

3 Basic algorithm

A solution is to enumerate all connected (k,SMP)-cores in the entire HIN G and select the top-r most semantically cohesive SICs. However, such exhaustive enumeration is computationally prohibitive. To address this, we propose a simple yet effective baseline, `Basic`, which constructs a small q-centric SMP-graph $G_{\text{SMP}} \subseteq G$ that contains all potential SICs, thereby significantly reducing the search space instead of the entire G. It is noteworthy that the smaller the G_{SMP} is, the faster the `Basic` runs.

Definition 11 (**q-centric SMP-graph**). *Given an HIN G, an SMP, and a query node q, we define the q-centric SMP-graph as $G_{\text{SMP}} \subseteq G$, a connected component containing q with all nodes connected via $\mathcal{P}^*$-edges, for any wildcard $\mathcal{P}^* \models$ SMP.*

Figure 2 (middle) vividly illustrates the constructed G_{SMP}, which is derived from the HIN in Fig. 1, with SMP = $\langle$Person, Song, Artist, 3$\rangle$ and $q = v_1$. This G_{SMP} contains 11 nodes of type Person, which are densely connected via $\mathcal{P}^*$-edges with $\mathcal{P}^*$ represents any one of the five meta-paths shown in Fig. 2 (left).

Lemma 1 *Given an HIN G, an SMP, and a query node q, any connected (k, SMP)-core containing q must be a subgraph of the q-centric SMP-graph $G_{SMP} \subseteq G$.*

Proof By Definition 11, G_{SMP} is the largest connected subgraph that contains q and in which all nodes are connected via $\mathcal{P}^*$-edges. Since a connected (k,SMP)-core of q requires every node to have at least k $\mathcal{P}^*$-neighbors and be reachable via $\mathcal{P}^*$-edges, it must be contained in the G_{SMP} of q.

Overview Algorithm 1 outlines the procedure of `Basic`. It operates based on the semantic monotonicity of r-SICS: the semantic cohesiveness of the top-r communities decreases as r increases. More precisely, as more SICs are extracted, new edge types would be introduced, increasing semantic diversity but reducing cohesiveness. With this in mind, we use a *semantic constraint set* to regulate the semantic scope of the search. Specifically, it progressively expands the semantic constraint set by adding new edge types. For each expanded semantic constraint set, `Basic` constructs a G_{SMP} using all edge types in it and performs core decomposition on G_{SMP} to identify candidate SICs. The process continues until the top-r SICs with the highest semantic cohesiveness are obtained.

https://doi.org/10.48130/ker-0026-0008

Algorithm 1. `Basic`.

Input: HIN G, SMP = $\langle A_t, A_a, S, \hat{l}\rangle$, q, $k \geq 0$, $r \geq 1$
Output: top-r SICs with highest semantic cohesiveness
1 `HSet` $\leftarrow \emptyset$; `SemSet` $\leftarrow \emptyset$; $H' \leftarrow \emptyset$;
2 `count` $\leftarrow 0$; // number of SICs
3 R $\leftarrow \{S_i \in \mathcal{R}\}$, in descending order of $W_{S_i \to S}$;
4 **while:** `count` $< r$ **do**
// Step (1): Semantic expansion
5 **if** |`SemSet`| = |R| **then**
6 **break**;
7 `SemSet` $\leftarrow$ `SemSet` $\cup$ R.poll();
// Step (2): G_{SMP} construction
8 $G_{\text{SMP}} \leftarrow$ `getSMPGraph`(G, SMP, q, `SemSet`);
// Step (3): SIC extraction
9 $H \leftarrow$ find a connected (k,SMP)-core of q from G_{SMP};
10 **if** H *does not exist* **then**
11 **continue**;
// Step (4): Quality evaluation
12 **if** $|V(H)| > |V(H')|$ **then**
13 `HSet` $\leftarrow$ `HSet` $\cup H$;
14 $H' \leftarrow H$;
15 `count` $\leftarrow$ `count` $+ 1$;
16 **return** `HSet`;

`Basic`. In Algorithm 1, we first initialize several key structures: an empty set `HSet` to store discovered r SICs, an empty semantic constraint set `SemSet` to control the semantic scope, a subgraph H' to store the previous SIC, and a counter `count` to record the number of SICs found so far (lines 1-2). Then, we rank all edge types in descending order of $W_{S_i \to S}$ and store them in a list R (line 3). After that, we iterate the following steps until the top-r SICs are found: (1) *Semantic expansion*. We progressively add new edge types (one by one) from R into `SemSet` to increase the semantic diversity. If `SemSet` already contains all semantics in R, the algorithm terminates, as no further SICs with lower semantic cohesiveness can be found (lines 5–7). (2) G_{SMP} *construction*. We construct G_{SMP} using the updated `SemSet` (line 8, detailed in Algorithm 2). (3) *SIC extraction*. We adopt the core decomposition[39] to iteratively remove nodes with less than k $\mathcal{P}^*$-neighbors until all target nodes have at least k $\mathcal{P}^*$-neighbors. The remaining connected component that contains q is a feasible SIC H (line 9). (4) *Quality evaluation*. We check if the newly discovered H satisfies the size monotonicity mentioned in the definition of r-SICS problem (lines 10–15).

We now focus on the detailed process of constructing G_{SMP} (the step (2) above), which is the key to the `Basic` algorithm.

G_{SMP} construction. The procedure of `getSMPGraph` is outlined in Algorithm 2 We first initialize a queue `Expand` = $\{q\}$ to store candidate nodes for expansion, an empty set `Visited` to keep track of visited nodes, and an empty graph G_{SMP} (line 1). Then, G_{SMP} is constructed by two steps: (1) *Neighbor retrieval*. For each node in `Expand`, invoke `FPN`(G, u, SMP, `SemSet`) detailed in Algorithm 3 to retrieve its $\mathcal{P}^*$-neighbors and $\mathcal{P}^*$-edges that formed by edge types from `SemSet`. (lines 3–4). (2) *Graph update*. We add the discovered $\mathcal{P}^*$-neighbors and edges to G_{SMP}, then add unvisited neighbors into `Expand` and move the visited nodes into `Visited` (lines 5–7). These steps are repeated until the `Expand` queue becomes empty.

- `FPN`(G, u, SMP, `SemSet`). In Algorithm 3, we initialize: an empty set $N^*(u)$ to store the $\mathcal{P}^*$-neighbors of a node u, an empty set $E^*(u)$ for $\mathcal{P}^*$-edges of u, a queue `NbrExpand` to track nodes to be expanded, and a `Path` to cache temporary path instances discovered during the search (lines 1–3). It then iterates the following to retrieve all valid $\mathcal{P}^*$-neighbors and $\mathcal{P}^*$-edges: (1) *Neighbor exploration*. We dequeue a node w from `NbrExpand` and retrieve all its neighbors from the original graph G (lines 5–7). (2) *Path extension*. If the edge type between w and its neighbor v is contained in `SemSet`, we extend the path p_{uw} with an edge e_{wv} to form a new path instance p_{uv} (lines 9–13). (3) *Neighbor update*. If v is of the target type A_t and p_{uv} matches the SMP, then we add v to the $\mathcal{P}^*$-neighbor set $N^*(u)$ and store p_{uv} in the $\mathcal{P}^*$-edge set $E^*(u)$. Otherwise, the length of p_{uv} is less than upper bound $\hat{l}$, we add v into `NbrExpand` and add p_{uv} into `Path` (lines 14–19). We repeat steps (1)–(3) until all expansion operations are completed and no nodes remain to be expanded.

Effectiveness analysis. We analyze the correctness and effectiveness of `Basic`. Specifically, we rigorously demonstrate that the algorithm can (1) correctly identify the H found in each iteration satisfying all constraints specified in Definition 10 (Theorem 1): query participation, structure cohesiveness and, maximality, and (2) correctly obtain the top-r SICs with the highest semantic cohesiveness, in accordance with the size and semantic monotonicity of the r-SICS problem (Theorem 2).

Theorem 1. *In each iteration, the subgraph H returned by* `Basic` *satisfies all SIC constraints.*

Proof The subgraph H returned by `Basic` is a (k,SMP)-core that contains the query node q, thus obviously satisfying the first two SIC constraints. For maximality, since we apply core-decomposition to find H, H must be the largest connected (k,SMP)-core in G_{SMP}. Suppose there exists a larger connected (k,SMP)-core satisfying the other two SIC constraints, then it must be outside G_{SMP}; otherwise, it will be identified as H by core-decomposition. However, this contradicts Lemma 1, that is, any (k,SMP)-core that contains q must be a subgraph of G_{SMP}. So, H is maximal.

Theorem 2 *The* `Basic` *algorithm can correctly discover the top-r SICs with the highest semantic cohesiveness, satisfying the size and semantic monotonicity of r-SICS problem.*

Proof For size monotonicity. In the `Basic` algorithm, each newly discovered H is compared with the previously found SIC H' to ensure that its size is larger, thereby fully satisfying the size monotonicity. For semantic monotonicity. In `Basic`, each expansion of the semantic constraint set `SemSet` introduces an additional edge type with lower semantic similarity. This means that the semantic cohesiveness of the new subgraph H obtained based on the updated `SemSet` cannot be higher than that of the previous H', which is entirely consistent with the semantic monotonicity.

Complexity analysis. The overall time of `Basic` is $O(r \times (|V_{\text{SMP}}| \times d^{\hat{l}} + |V_{\text{SMP}}| + |E_{\text{SMP}}|))$, where r is the number of SICs to be found, $|V_{\text{SMP}}|$ is the number of target nodes in G_{SMP}, d is the

Algorithm 2. `getSMPGraph`(G, SMP, q, `SemSet`).

Input: HIN G, SMP = $\langle A_t, A_a, S, \hat{l}\rangle$, q, `SemSet`
Output: q-centric SMP-graph G_{SMP}
1 `Expand` $\leftarrow q$; `Visited` $\leftarrow \emptyset$; $G_{\text{SMP}} \leftarrow \emptyset$;
2 **while:** `Expand` $\neq \emptyset$ **do**
// Step (1): Neighbor retrieval
3 $u \leftarrow$ `Expand`.poll();
4 $\langle N^*(u), E^*(u)\rangle \leftarrow$ `FPN`(G, u, SMP, `SemSet`);
// Step (2): Graph update
5 $G_{\text{SMP}} \leftarrow G_{\text{SMP}} \cup \langle N^*(u), E^*(u)\rangle$;
6 `Visited` $\leftarrow$ `Visited` $\cup \{u\}$;
7 `Expand` $\leftarrow$ `Expand` $\cup (N^*(u) \setminus$ `Visited`$)$;
8 **return** G_{SMP};

average node degree in original G, $\hat{l}$ is length bound and $|E_{SMP}|$ is the number of $\mathcal{P}^*$-edges in G_{SMP}. The main computational cost comes from `getSMPGraph` (Algorithm 2), of which each node calls `FPN` (Algorithm 3), which has a worst-case complexity of $O(d^{\hat{l}})$ because each node may generate up to $d^{\hat{l}}$ path instances matching the SMP. Subsequently, the core-decomposition is applied on G_{SMP}, with time of $O(|V_{SMP}|+|E_{SMP}|)$ in the worst case.

4 Optimized algorithm

The `Basic` algorithm requires reconstructing G_{SMP} from scratch each time the semantic scope is expanded with a new edge type, resulting in redundant computations. To address this, we propose the *Optimized Algorithm with Binary Semantic Search* (`Opt-BSS`) in section 4.1. It expands the semantic scope in batches and constructs a maximal G_{SMP} encompassing all involved edge types. It then employs a semantic-level binary search to peel smaller, inherent G_{SMP} from the maximal one, avoiding costly reconstruction from scratch. Building on this, section 4.2 introduces the *Optimized Algorithm with Recording* (`Opt-Record`), which accelerates SICs search from G_{SMP} by reusing intermediate information during binary search.

4.1 Optimized with binary semantic search

Overview The basic idea of `Opt-BSS` is two-fold. First, it adds edge types to `SemSet` in batches of size b (the effect of b is studied in section 6.4), therefore constructing a comprehensive G_{SMP} that is likely to contain the top-r SICs, avoiding repeated reconstruction of G_{SMP} for each edge type. Second, it adopts a semantic binary search to efficiently locate the top-r SICs from the above G_{SMP}. Specifically, it divides edge types into a strong and weak semantic parts based on their $W_{S_i \to S}$. If a smaller $G'_{SMP} \subseteq G_{SMP}$ exists within the strong semantic part and SICs can be found from G'_{SMP}, we continue the binary search within the strong semantic part; otherwise, we search for SICs in another smaller $G'_{SMP} \subseteq G_{SMP}$ derived from the weak semantic part. There may exist multiple SICs within a single batch. Therefore, after one SIC is identified, the binary search process described above is executed again on the remaining semantics in the batch until the top-r SICs are completely found. Alternatively, if fewer than r SICs are found in the current batch, we expand `SemSet` with another batch of edge types and repeat the above.

`Opt-BSS`. Algorithm 4 outlines `Opt-BSS`. We initialize an empty `HSet` for top-r SICs, an empty semantic constraint set `SemSet`, and an empty H' for the SIC with the highest semantic cohesiveness so far (line 1). All edge types are stored in a list R in descending order of their $W_{S_i \to S}$ (line 2). We add edge types in batches of size b from R to `SemSet`, following their ranking. To facilitate this process, we maintain a sliding window $[U,D]$ to track the current batch, with U and D initialized as 1 and b, indicating the first and the b-th edge type in R. `Opt-BSS` then iterates the following steps until the top-r SICs are searched. (1) *Batch semantic expansion*. We continuously slide the window (i.e., update U and D) to add edge types in batches (lines 5–10). (2) G_{SMP} *construction*. Using the current `SemSet`, we construct a G_{SMP} by applying Algorithm 2 (line 11). (3) *SIC extraction*. We apply the same method as in `Basic` to extract a feasible SIC H from current G_{SMP} (line 12). (4) *Binary semantic search*. If H is larger than

Algorithm 3. `FPN`(G,u, SMP, `SemSet`).

Input: HIN G, node u, SMP = $\langle A_t, A_a, S, \hat{l}\rangle$, q, `SemSet`
Output: $\mathcal{P}^*$-neighbors and edges of node u $\langle N^*(u), E^*(u)\rangle$
1 $N^*(u) \leftarrow \emptyset$; $E^*(u) \leftarrow \emptyset$;
2 `NbrExpand` $\leftarrow u$;
3 `Path` $\leftarrow p_{uu}$;
4 **while:** `NbrExpand` $\neq \emptyset$ **do**
// `Step (1): Neighbor exploration`
5 $w \leftarrow$ `NbrExpand`.poll();
6 $p_{uw} \leftarrow$ `Path`.poll();
7 `Neighbor`(w) $\leftarrow$ w's neighbors in G;
8 **while:** `Neighbor`(w) $\neq \emptyset$ **do**
// `Step (2): Path extension`
9 $v \leftarrow$ `Neighbor`(w).poll();
10 **if** `SemSet`.*contains*($\psi(e_{wv})$) **then**
11 $p_{uv} \leftarrow p_{uw} \cup e_{wv}$;
12 **if** $u = v$ **then**
13 continue;
// Step (3): Neighbor update
14 **if** p_{uv}.*length*() $\leq \hat{l}$ ***and*** $\phi(v) = A_t$ ***and*** $\phi(x) = A_a$ *(x is a node in* p_{uv}*)* **then**
15 $N^*(u) \leftarrow N^*(u) \cup v$;
16 $E^*(u) \leftarrow E^*(u) \cup p_{uv}$;
17 **else if** p_{uv}.*length*() $< \hat{l}$ **then**
18 `NbrExpand` $\leftarrow$ `NbrExpand` $\cup v$;
19 `Path` $\leftarrow$ `Path` $\cup p_{uv}$;
20 **return** $\langle N^*(u), E^*(u)\rangle$;

Algorithm 4. `Opt-BSS`.

Input: HIN G, SMP = $\langle A_t, A_a, S, \hat{l}\rangle$, q, $k \geq 0$, $r \geq 1$, $b > 1$
Output: top-r SICs with highest semantic cohesiveness
1 `HSet` $\leftarrow \emptyset$; `SemSet` $\leftarrow \emptyset$; $H' \leftarrow \emptyset$;
2 R $\leftarrow \{S_i | S_i \in \mathcal{R}\}$, sorted in descending order;
3 $U \leftarrow 1$;
4 **while:** $U <$ |R| **do**
// `Step (1): Batch semantic expansion`
5 **if** $U + b <$ |R| **then**
6 $D = U + b$;
7 **else**
8 $D =$ |R|;
9 **for** $i \leftarrow U$ *to* D **do**
10 `SemSet` $\leftarrow$ `SemSet` $\cup$ R.get(i);
// `Step (2): Construct` G_{SMP}
11 $G_{SMP} \leftarrow$ `getSMPGraph`(G, SMP, q, `SemSet`);
// `Step (3): SIC extraction`
12 $H \leftarrow$ find a connected (k,SMP)-core of q from G_{SMP};
// `Step (4): Binary semantic search`
13 **if** H *does not exist* **then**
14 $U \leftarrow D + 1$;
15 **continue**;
16 **else**
17 **if** $|V(H)| > |V(H')|$ **then**
18 `HSet` $\leftarrow$ `HSet` $\cup$ `BSS`(H', G_{SMP}, U, D, R, r, |HSet|, q);
19 **if** |HSet| $= r$ **then**
20 **break**;
21 $H' \leftarrow H$;
22 $U \leftarrow D + 1$;
23 **else**
24 $U \leftarrow D + 1$;
25 **continue**;
26 **return** HSet;

https://doi.org/10.48130/ker-0026-0008

H', this indicates that the current G_{SMP} contains new SICs at different semantic levels from that of H' and they can be found by calling BSS (detailed in Algorithm 5). After the current round of binary semantic search ends, these new SICs are added to HSet, U is updated to $D+1$, and H' is updated to the current H. (lines 13-26).

BSS. Algorithm 5 outlines the process of binary semantic search in a G_{SMP}. First, it initializes a left (right) boundary left (right) to the highest (lowest) semantic rank U (D), which are used to control the range of binary search, an empty result set HSet, and an empty local semantic constraint set for BSS SemSet_BSS (lines 1 and 2). It then performs binary semantic search within the current range [left,right] to find the top-r SICs. (1) *Midpoint selection*. We calculate the middle rank mid based on the left and right. Edge types with ranks greater than or equal to mid are considered strong semantic part, and these edge types are added to the local SemSet_BSS, which is reset to empty before update to ensure accurate edge type acquisition. (lines 6–11). (2) G'_{SMP} *construction*. Next, a smaller G'_{SMP} is constructed from the original G_{SMP} by removing edges that excluded from SemSet_BSS (line 12). (3) *SIC extraction*. Then, we apply core-decomposition to find a connected (k,SMP)-core H containing q from G'_{SMP} (line 13). (4) *SIC refinement*. If the new SIC H is larger than the previous H', BSS continues binary search in the strong semantics part to check for a more cohesive and larger SIC. If mid equals left, H is the best in this part, so we update H' and HSet, increments count, and adjusts semantic boundaries. Otherwise, we continue binary search with the right boundary as mid–1 (lines 14–23). If no better H is found, the search next try to select more semantics from weak semantic part by setting left to mid+1. When left exceeds right, we check if edges ranked from left onward can yield a better SIC. If so, it updates for H', HSet, count, and boundaries (lines 24–30). Notably, when searching for the i+1$^{\text{th}}$ SIC, boundary left is set to mid+1, where mid is the midpoint when finding i^{th} SIC. Meanwhile, boundary right is reset to the lowest semantic rank D (line 20). We perform a semantic binary search within the updated range to obtain the i+1$^{\text{th}}$ SIC. By analogy, we repeat steps (1)–(4) until the top-r SICs are found or the binary semantic search can no longer proceed.

Algorithm 5. BSS(H', G_{SMP}, U, D, R, r, count, q).

Input: H', G_{SMP}, $U>0$, $D>0$, R, $r\geq 1$, count ≥ 0, q
Output: top-r SICs with highest semantic cohesiveness

1 left $\leftarrow U$; right $\leftarrow D$;
2 HSet $\leftarrow \emptyset$; SemSet $\leftarrow \emptyset$;
3 **while** left $\leq$ right **do**
// Termination check
4 **if** count = r **then**
5 **break**;
6 **if** count = 0 **then**
7 $H' \leftarrow \emptyset$;
// Step (1): Midpoint selection
8 mid = left + $\lfloor$(right − left)/2$\rfloor$;
9 SemSet_BSS $\leftarrow \emptyset$;
10 **for** $i \leftarrow 1$ *to* mid **do**
11 SemSet_BSS $\leftarrow$ SemSet_BSS $\cup$ R.get(i);
// Step (2) and (3)
12 $G'_{\text{SMP}} \leftarrow$ find a smaller G'_{SMP} using SemSet_BSS;
13 $H \leftarrow$ find a connected (k,SMP)-core of q from G'_{SMP};
// Step (4): SIC refinement
14 **if** H *exists and* $|V(H)| > |V(H')|$ **then**
15 **if** mid = left **then**
16 $H' \leftarrow H$;
17 count = count + 1;
18 HSet $\leftarrow$ HSet $\cup H$;
19 left $\leftarrow$ mid + 1;
20 right $\leftarrow D$;
21 **continue**;
22 **else**
23 right = mid − 1;
24 **else**
25 left = mid + 1;
26 **if** left > right **then**
27 SemSet_BSS $\leftarrow$ SemSet_BSS $\cup$ R.get(left);
28 repeat lines 12-13;
29 **if** H *exists* **then**
30 repeat lines 16–21;
31 **return** HSet;

Effectiveness analysis. Opt-BSS can correctly identify SICs satisfying constraints in Definition 10, as each SIC is obtained in the same manner as Basic. Moreover, Since the semantic boundaries of binary search are adjusted to ensure that each newly found SIC satisfies the semantic monotonicity, and the size of each SIC is evaluated in line 17 to guarantee the size monotonicity. Therefore, Opt-BSS is capable of returning the top-r SICs for r-SICS problem.

Complexity analysis. The overall time of BSS-Opt is $O(\frac{|\mathcal{R}|}{b}\cdot(|V_{\text{SMP}}|\cdot d^{\hat{l}}+|V_{\text{SMP}}|+|E_{\text{SMP}}|+r\cdot(\log b\cdot|V_{\text{SMP}}|+|V'_{\text{SMP}}|+|E'_{\text{SMP}}|)))$, where $|\mathcal{R}|$ is the number of edge types, b is the batch size, $|V_{\text{SMP}}|$ ($|V'_{\text{SMP}}|$) and $|E_{\text{SMP}}|$ ($|E'_{\text{SMP}}|$) are the number of target nodes and $\mathcal{P}^*$-edges in G_{SMP} (G'_{SMP}). In the worst case, it performs $\frac{|\mathcal{R}|}{b}$ batches of semantic expansions. In each batch, it constructs a new G_{SMP} by Algorithm 2 and performs a core-decomposition on it, which costs $O(|V_{\text{SMP}}|\cdot d^{\hat{l}}+|V_{\text{SMP}}|+|E_{\text{SMP}}|)$ time. In practice, when b is appropriately set, only 1–2 batches are needed to obtain the top-r SICs. Within each batch, it invokes Algorithm 5 to find an SIC with at most $\log b$ iterations, which costs $O(\log b\cdot|V_{\text{SMP}}|+|V'_{\text{SMP}}|+|E'_{\text{SMP}}|)$ time. Since at most r SICs are identified from a single G_{SMP}, the time for one batch is $O(r\cdot(\log b\cdot|V_{\text{SMP}}|+|V'_{\text{SMP}}|+|E'_{\text{SMP}}|))$. Thus, the total time of BSS-Opt is $O(\frac{|\mathcal{R}|}{b}\cdot(|V_{\text{SMP}}|\cdot d^{\hat{l}}+|V_{\text{SMP}}|+|E_{\text{SMP}}|+r\cdot(\log b\cdot|V_{\text{SMP}}|+|V'_{\text{SMP}}|+|E'_{\text{SMP}}|)))$.

4.2 Optimized algorithm with recording

In Section 4.1, the Opt-BSS algorithm sets the search boundary for the $(i+1)^{th}$ SIC as [mid$_i$ + 1, D]. This is a crude boundary selection, which causes a large number of semantics to be repeatedly added to SemSet_BSS for SIC discovery, thereby significantly reducing search efficiency. To address this issue, the Opt-Record algorithm proposed in this section records intermediate information during each binary semantic search and reuses it in subsequent searches, thereby effectively avoiding repeated calculations.

Overview The key to implementing Opt-Record lies in identifying the information that should be recorded. To this end, we start from Theorem 3, first rigorously proving an SIC constructed from a SemSet_BSS containing highly similar semantics is necessarily contained within an SIC formed by lower semantic similarity. Based on this, it is no longer necessary to use a large range with the right boundary is D for searching SICs; instead, the search can be confined by the already discovered SICs formed from SemSet_BSS containing lowly similar semantics. We thus design a global map findedH to record the discovered SICs and their corresponding semantics to implement the recorded binary semantic search.

Theorem 3 *Let* $\{H_1,\ldots,H_r\}$ *be* r *SICs of the same query node* q *with semantic cohesiveness from high to low. Then, for any* $1 \le i < r$, *we have* $H_i \subsetneq H_{i+1}$.

Proof Let node $v \in V_{H_i}$ but $v \notin V_{H_{i+1}}$ for $i \in (0, r-1]$. By Definitions 10 and 6, the semantic cohesiveness between any two nodes in V_{H_i} satisfies $W_{\mathcal{P}_{uv}\to\text{SMP}} \ge W_{H_i\to\text{SMP}}$. Since we assume $W_{H_i\to\text{SMP}} \ge W_{H_{i+1}\to\text{SMP}}$, $W_{\mathcal{P}_{uv}\to\text{SMP}} \ge W_{H_{i+1}\to\text{SMP}}$ holds. According to the maximality constraint in Definition 10, H_{i+1} should be the largest connected (k,SMP)-core at the semantic level $W_{H_{i+1}\to\text{SMP}}$. However, by including v via $\mathcal{P}_{uv}$ would enlarge H_{i+1}'s size while keep $W_{H_{i+1}\to\text{SMP}}$ unchanged, contradicting the maximality of H_{i+1}. So, v must also belong to H_{i+1}, implying $H_i \subsetneq H_{i+1}$.

`Opt-Record`. `Opt-Record` is detailed in Algorithm 6. Compared with the `Opt-BSS` algorithm, `Opt-Record` introduces an additional map `findedH` to record the SICs discovered during the search process and their corresponding semantic levels, i.e., edge type count in the semantic constraint set when SIC was constructed. `Opt-Record` shares the same framework as Algorithm 4, iteratively executing four steps until the top-r communities are found: (1) Batch semantic expansion. (2) G_{SMP} construction. (3) SIC extraction. (Details of Steps 1–3 are provided in Algorithm 4). (4) Binary semantic search with recording. we perform a semantic binary search in H via BSS-Record (in Algorithm 7). After `BSS-Record` finishes, the newly discovered SICs are added to `HSet`. If top-r SICs are not found, we update U to $D+1$, set H' to the current H, and proceed to the next iteration until the top-r SICs are found(lines 7–14).

`BSS-Record`. Algorithm 7 describes binary semantic search with record. Initially, we set left (right) boundary `left` (`right`) as U (D), which control the semantic range during the binary search, and result set `HSet`, semantic constraint set `SemSet_BSS` as empty. An empty stack `lastRight` is used to record previous right boundaries, and an empty set `SemSizeSet` to store SIC-formable semantic ranks (lines 1–3). The algorithm proceeds as follows: (1) Midpoint selection. Compute middle semantic rank to generate the corresponding semantic constraint set `SemSet_BSS` (detailed in Algorithm 5). (2) SIC extraction. By Theorem 3, SICs with smaller semantic range are necessarily contained within larger ones. Thus, we retrieve the SIC that is larger than but most similar to the current semantic range from `findedH` and denoted as `lastH`. Then, edges in `lastH` that violate the current semantic constraints are removed to obtain the SIC H that conforms to `SemSet_BSS` (lines 6–10). (3) SIC refinement. Similar to Algorithm 5, if the size of H is larger than H' and `mid` exactly equals `left`, the algorithm updates H', `HSet`, `count`, and the left and right boundaries (detailed in Algorithm 5). Notably, if the left boundary exceeds the right boundary, the right boundary is extended to the previously stored right boundary in `lastRight`. If `mid` does not equal `left`, the binary search continues in the strong semantic segment to find an SIC with higher semantic cohesiveness. At this point, `findedH` records the current H, `lastRight` records the current right boundary, `SemSizeSet` stores the current semantic rank, and the right boundary is updated to `mid`-1 (lines 19–23). If no better H is found, the search shifts to the weak semantic segment, continuing by updating the left and right boundaries. We then check whether the most semantically cohesive SIC in the weak semantic segment has already been searched (i.e., whether the semantic rank `mid`+1 exists in `SemSizeSet`) (lines 25–27). If so, `HSet`, H', `count`, and the boundaries are updated accordingly (detailed in Algorithm 5). Steps 1–3 are repeated until the top-r SICs are found or the binary semantic search can no longer proceed.

Algorithm 6. `Opt-Record`.

Input: HIN G, SMP $= \langle A_t, A_a, S, \hat{I}\rangle$, query node q, integer $k \ge 0$, integer $r > 0$, integer $b > 1$
Output: SICs with top-r semantic cohesiveness

1 `HSet` $\leftarrow \emptyset$; `SemSet` $\leftarrow \emptyset$; $H' \leftarrow \emptyset$;
2 `R` $\leftarrow \{S_i | S_i \in \mathcal{R}\}$, sorted in descending order;
3 $U \leftarrow 1$; `count` $\leftarrow 0$; //
4 **while:** $U <$ |`R`| **do**
// `Step (1)-(3)`
5 lines 5-12 in `Opt-BSS` (Algorithm 4)
// `Step (4): Binary semantic search with recording`
6 **if** H *does not exist* **then**
7 lines 14-15 in `Opt-BSS` (Algorithm 4)
8 **else**
9 **if** $|V(H)| > |V(H')|$ **then**
10 `findedH`.put(`mid`, H);
11 `HSet` $\leftarrow$ `HSet` $\cup$ `RecordBSS`(H', U, D, `R`, r, `count`, q, `findedH`);
12 lines 19-22 in `Opt-BSS` (Algorithm 4)
13 **else**
14 lines 24-25 in `Opt-BSS` (Algorithm 4)
15 **return** `HSet`;

Algorithm 7. `BSS-Record`(H', U, D, `R`, r, `count`, q, `findedH`).

Input: previously discovered community H', integer $U > 0$, integer $D > 0$, ordered list of edge types `R`, integer $r > 0$, integer `count` ≥ 0, query node q
Output: SICs with top-r semantic cohesiveness

1 `left` $\leftarrow U$; `right` $\leftarrow D$;
2 `HSet` $\leftarrow \emptyset$; `SemSet_BSS` $\leftarrow \emptyset$;
3 `lastRight` $\leftarrow \emptyset$; `SemSizeSet` $\leftarrow \emptyset$;
4 **while:** `left` $\le$ `right` **do**
// `Step (1): Midpoint selection`
5 lines 4–11 in BSS (Algorithm 5)
// `Step (2): SIC extraction`
6 **if** `right = D` **then**
7 `lastH` = `findedH`.get(`right`)
8 **else**
9 `lastH` = `findedH`.get(`right + 1`)
10 $H \leftarrow$ find a connected (k,SMP)-core that satisfies `SemSet_BSS` in `lastH` containing q;
// `Step (3): SIC refinement`
11 **if** H *exists and* $|V(H)| > |V(H')|$ **then**
12 **if** `mid` = `left` **then**
13 `count` = `count` + 1;
14 lines 16–20 in BSS (Algorithm 5)
15 **if** `left` > `right` **then**
16 **if** `lastRight`.*isEmpty()* **then**
17 **break**;
18 `right` $\leftarrow$ `lastRight`.pop();
19 **else**
20 `findedH`.put(`mid`, H);
21 `lastRight`.push(`right`);
22 `right` = `mid` − 1;
23 `SemSizeSet` $\leftarrow$ `SemSizeSet` $\cup$ `mid`
24 **else**
25 `left` = `mid` + 1;
26 **if** `SemSizeSet`.*contains(*`mid` + 1*)* **then**
27 lines 16–18 in BSS (Algorithm 5)
28 repeat lines 15–18
29 **return** `HSet`;

https://doi.org/10.48130/ker-0026-0008

Effectiveness analysis. Next, we thoroughly analyze the correctness and effectiveness of the `Opt-Record` algorithm from two perspectives. First, `Opt-Record` can accurately identify SICs that satisfy all the constraints in Definition 10. According to Theorem 3, under the maximality and monotonicity constraints of our r-SICS formulation, an SIC with stronger semantic cohesiveness is contained within a broader SIC with weaker semantic cohesiveness. Therefore, `Opt-Record` can start from a previously discovered broader SIC, remove edges that do not satisfy the current semantic requirements, and then apply core decomposition to recover the maximal connected (k,SMP)-core containing q. First, `Opt-Record` can effectively recover the maximal connected (k,SMP)-core satisfying the current semantic constraints. Second, similar to `Opt-BSS`, `Opt-Record` adjusts the boundaries of the semantic binary search to ensure that each newly discovered SIC satisfies the semantic monotonicity condition, while also evaluating the size of each SIC to guarantee that the top-r results satisfy the size monotonicity requirement. Therefore, `Opt-Record` can correctly obtain the top-r SICs with the highest semantic cohesiveness as required by the r-SICS problem (Problem 2.3). In summary, the `Opt-Record` algorithm can correctly identify the top-r SICs that fulfill all the requirements of the r-SICS problem.

Complexity analysis. The outer loop of `Opt-Record` proceeds in batches, where each batch processes b semantic levels. Therefore, the maximum number of iterations is $\frac{|\mathcal{R}|}{b}$. For each batch, the time cost can be divided into three parts. (1) Construction of G_{SMP}: the cost is $O(|V_{\text{SMP}}|\cdot d^l)$, where $|V_{\text{SMP}}|$ is the number of nodes in G_{SMP} and d^l denotes the semantic expansion factor. (2) SIC extraction: the cost is $O(|V_{\text{SMP}}|+|E_{\text{SMP}}|)$. (3) Binary semantic search with recording (`Opt-Record`): the number of binary steps is $O(\log b)$. Each refinement requires processing the previously discovered community H, costing $O(|V_{\text{H}}|+|E_{\text{H}}|)$. Hence, the total cost of this step is $O((|V_{\text{H}}|+|E_{\text{H}}|)\cdot\log b)$. Combining the above, the overall time complexity of `Opt-Record` is $O(\frac{|\mathcal{R}|}{b}(|V_{\text{SMP}}|\cdot d^l+(|V_{\text{SMP}}|+|E_{\text{SMP}}|)+(|V_{\text{H}}|+|E_{\text{H}}|)\log b))$. Since H is always a subgraph of G_{SMP}, we have $|V_{\text{H}}|\le|V_{\text{SMP}}|$ and $|E_{\text{H}}|\le|E_{\text{SMP}}|$. Thus, the time complexity can be further simplified to $O(\frac{|\mathcal{R}|}{b}(|V_{\text{SMP}}|\cdot d^l+(|V_{\text{SMP}}|+|E_{\text{SMP}}|)\log b))$.

5 Parallel algorithm acceleration

Since the $\mathcal{P}^*$-neighbor expansion during the G_{SMP} construction is completely independent with no data dependencies among them, a simple yet highly effective acceleration approach is to adopt a parallel expansion via multithreading. Specifically, once a new $\mathcal{P}^*$-neighbors is identified, it is assigned to an independent thread to search for its subsequent $\mathcal{P}^*$-neighbors. This speeds up G_{SMP} construction, thereby reducing the overall runtime.

Sequential expansion. As shown in Fig. 4 (left), the sequential expansion begins from the initial node v_1, recording its neighbors v_2, v_3, and v_4, and then visits them sequentially to discover their neighbors v_6, v_7, v_8, and v_5. The expansion continues iteratively with newly found nodes until all nodes are traversed. Here, completing the entire traversal requires 12 steps (the sequence of steps is associated with edges). For a large HIN with numerous $\mathcal{P}^*$-neighbors, sequential expansion inevitably leads to linearly increasing construction time and thus results in significantly low efficiency.

Parallel expansion. In contrast, the parallel $\mathcal{P}^*$-neighbor expansion shown in Fig.4 (right) starts with the initial node as the first $\mathcal{P}^*$-neighbor, assigned to Thread 1 for discovering its $\mathcal{P}^*$-neighbors v_6, v_7, v_8, and v_5. While Thread 1 is expanding from v_6, idle Threads 2 and 3 can simultaneously process the expansion from v_7 and v_5, respectively. This parallel expansion completes the full traversal in just seven steps—nearly doubling the efficiency compared to the sequential expansion. Experimental results on real-world datasets (details in section 6) demonstrate that the G_{SMP} construction with parallel $\mathcal{P}^*$-neighbor expansion can significantly improve the runtime efficiency. For the million-scale DBpedia dataset, this parallel construction improves the overall efficiency of `Basic` by 5× and enhances that of `Opt-Record` by 3× on average.

Remark. Although a fine-grained parallelization could be applied to the internal steps of the $\mathcal{P}^*$-neighbor expansion, it would introduce significant synchronization overhead. Therefore, we consider the search for a $\mathcal{P}^*$-neighbor as an atomic procedure and parallelize the G_{SMP} construction at this level.

6 Experiment

We conducted an experimental study to answer four questions. All experiments were run on a Linux Server with 3.7 GHz CPU and 128 GB memory. Our code and datasets are available at the GitHub repository[40].

Q1: What is the effectiveness of the algorithms in practice? (**section 6.2**)
Q2: How do algorithms perform in terms of efficiency? (**section 6.3**)
Q3: How sensitive are the proposed algorithms to various parameter settings? (**section 6.4**)
Q4: How does our algorithms find the top-r SICs in semantic-rich HINs? (**section 6.5**)

6.1 Experimental setup

Datasets. Table 1 summarizes four million-scale semantic-rich HINs. (1) Wikidata[41] is a collaborative knowledge base that supports Wikimedia projects. (2) Freebase[42] is a knowledge graph derived from online wiki contributions. (3) YAGO[43] involves information

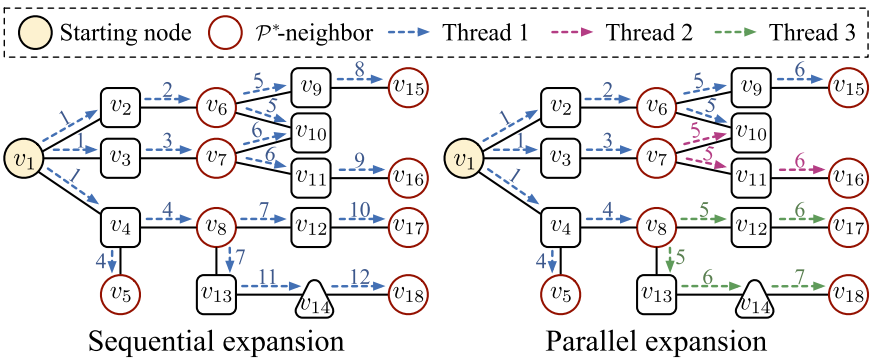

Fig. 4 Parallel $\mathcal{P}^*$-neighbor expansion.

Table 1. Statistics of datasets.

Datasets	Nodes	Edges	Node types	Edge types
Wikidata[41]	3,335,203	10,678,827	12,742	818
Freebase[42]	3,733,753	10,998,482	3,227	3,897
YAGO[43]	4,295,825	11,413,472	222,887	149
DBpedia[10]	7,374,882	22,113,471	432	627

from WordNet and GeoNames. (4) DBpedia[10] is an open-domain linked data built from Wikipedia.

Methods. (1) **Basic** is proposed in section 3. (2) **Opt−BSS** is the first optimized algorithm with binary semantic search proposed in section 4.1. (3) **Opt−Record** is the second optimized algorithm built atop Opt−BSS in section 4.2. It enhances G_{SMP} construction efficiency by caching the intermediate SICs, thereby reducing the search space for binary semantic search and avoiding redundant computations. We apply the parallel $\mathcal{P}^*$-neighbor expansion to (1)–(3) to form three variants (4) **Basic+P**, (5) **Opt−BSS+P**, and (6) **Opt−Record+P**. Since the problem of r-SICS on semantic-rich HINs has not been studied before, there are no existing methods that are fully aligned with our problem formulation. Therefore, we compare our methods with several representative and closely related community search baselines, and adapt them to our setting in a best-effort manner so that they are as aligned as possible under their own input forms. We compared our methods with three representative baselines considering node attributes: (7) ***r*KACS**[13], (8) **VAC**[4], and (9) **KC**[5]. Here, we follow the same setting as rKACS to assign attributes to a node as the types of its adjacent edges, thus building the connection between relational semantics and node attributes. rKACS returns top-r communities similar to the given query attributes, while VAC and KC can only return one community. In this setting, the higher attribute similarity of H to the query attributes indicates a higher relational semantic cohesiveness. In addition, we further compare our methods with a classic community search baseline on heterogeneous graphs, (10) **FastOnline**[15]. FastOnline supports the use of multiple predefined symmetric meta-paths within a single query. To adapt it to our setting, we use the symmetric meta-path associated with each SIC layer in the top-r results as the predefined meta-path for the corresponding query.

Queries. We generate 200 queries per dataset. For methods (1)–(6), we randomly select a query node q and use its type as A_t. Then, we randomly select a neighbor u of q, use its type as A_a, and take the edge type between q and u as the relational semantics S. We set the meta-path length bound $\hat{l} \leq 4$ as longer meta-paths tend to weaken relational strength[44–46]. Finally, a query is formed as q with SMP $= \langle A_t, A_a, R, \hat{l} \rangle$. For methods (7)–(9), we construct query attributes by collecting the edge types appearing in the top-r SICs returned for query node q, so as to provide these baselines with sufficiently informative and fair inputs under their own settings. In this way, the constructed query attributes can reflect semantic intentions similar to those in our problem formulation. As default, we set $r = 3$ and $\hat{l} = 3$.

Metrics. We evaluate effectiveness from three semantic perspectives: semantic diversity (NMP), semantic consistency and quality (ALSD and SC), and comparative semantic evaluation (AAS). (1) Number of Meta-Paths (**NMP**) is the number of distinct symmetric and asymmetric meta-paths within a community H. A higher average NMP across the top-r communities indicates richer semantic diversity. (2) Adjacent-Layer Semantic Difference (**ALSD**) measures the semantic discrepancy between adjacent communities in the top-r results, as $\text{ALSD}(H,r) = \frac{1}{r-1}\sum_{i\in[1,r-1]}(1-\frac{|\mathcal{R}_i \cap \mathcal{R}_{i+1}|}{|\mathcal{R}_i \cup \mathcal{R}_{i+1}|})$ ($\mathcal{R}_i$ is the edge types of the i-th community). A lower ALSD reflects stronger semantic continuity across the top-r communities with respect to the given SMP. (3) Semantic Cohesiveness **SC**, as defined in Definition 6. A higher average SC over the top-r communities indicates better semantic quality. We also adopt a metric from the evaluation framework of rKACS to assess effectiveness. (4) Average Attribute Similarity (**AAS**) measures how H is similar to query attributes Q_W, as $\text{AAS}(H,Q_W) = \frac{1}{|V(H)|}\sum_{u\in V(H)}\frac{|A(u)\cap Q_W|}{|A(u)\cup Q_W|}$. Higher AAS indicates better quality. Moreover, we evaluate efficiency via response time.

6.2 Effectiveness evaluation

Since our methods (1)–(6) yield identical results for the same query, their results are reported collectively in Tables 2 and 3.

NMP results. In Tables 2 and 3, the first column shows that ours consistently outperforms all baselines across all datasets, achieving the highest semantic diversity. This indicates that the SICs identified by us encompass more semantically similar meta-paths and capture richer multi-level relational semantics.

Table 2. Effectiveness evaluation on Wikidata and Freebase.

Method	Wikidata				Freebase			
	NMP	ALSD	SC	AAS	NMP	ALSD	SC	AAS
Ours (1)–(6)	**34**	**20.6%**	**0.844**	**0.291**	**34.5**	**21.5%**	**0.802**	0.186
*r*KAC	12.25	55.6%	0.576	0.181	15.5	50.6%	0.649	**0.209**
KC	15.7	–	0.413	0.101	21	–	0.294	0.082
VAC	3.2	–	0.558	0.081	4.5	–	0.504	0.105
FastOnline	5.3	–	–	0.124	13.7	–	–	0.163

Boldface indicates the best result, and underlining indicates the second-best result.

Table 3. Effectiveness evaluation for YAGO and DBpedia.

Method	YAGO				DBpedia			
	NMP	ALSD	SC	AAS	NMP	ALSD	SC	AAS
Ours (1)–(6)	**26**	**16.3%**	**0.898**	**0.226**	**77.75**	**26.1%**	**0.877**	**0.149**
*r*KAC	11.3	46.6%	0.543	0.128	13.17	73.9%	0.578	0.118
KC	7	–	0.503	0.115	22.8	–	0.639	0.131
VAC	2	–	0.524	0.111	2.75	–	0.571	0.116
FastOnline	8.2	–	–	0.142	18.3	–	–	0.105

Red values indicate the best results, while blue underlined values indicate the second-best results.

https://doi.org/10.48130/ker-0026-0008

ALSD results. In Tables 2 and 3 (2nd column), our methods achieve an average ALSD of 20.1%, which is lower than 56.7% of *r*KACS. This clearly indicates that our methods ensures consistently high semantic coherence across r communities with small deviation from the given SMP. In contrast, the high ALSD of *r*KACS suggests potential semantic discontinuity among communities. ALSD is not applicable to VAC and KC, as they return only one community. Moreover, since FastOnline adopts the symmetric meta-paths corresponding to each SIC layer as the semantics in each search, its semantic modeling is consistent with ours at this level. As a result, FastOnline yields the same ALSD values as our method, making this metric not directly comparable between the two approaches.

SC results. Tables 2 and 3 (3rd column) confirm our method's superior SC across all datasets. This strongly suggests that baselines may include semantically irrelevant meta-paths in the results, which consequently leads to lower SC. This is mainly because they optimize the similarity of individual edge types (i.e., node attributes), while we optimize the similarity of the entire meta-path. Similarly to ALSD, FastOnline is also not comparable under the SC metric.

AAS results. In Tables 2 and 3 (last column), our methods outperforms all baselines on three datasets and attains competitive results on Freebase. This clearly indicates strong attribute similarity between the communities and the query attributes. As we know, node attributes are typically derived from the types of their adjacent edges, and a higher AAS reflects greater relational semantic cohesiveness.

6.3 Efficiency evaluation

Efficiency comparison with baseline. Figure 5 further compares our methods with *r*KACS and FastOnline, two representative community search baselines on heterogeneous graphs. Since only these methods are capable of returning the top-r communities, we focus on their runtime performance. Across all datasets, our fastest method, Opt-Record + P, consistently achieves lower runtime than both baselines. Specifically, Opt-Record + P runs **6.4**× faster than *r*KACS on average, with notable speedups on DBpedia (**11.6**×) and YAGO (**7.9**×). Even without parallelism, Opt-Record still outperforms *r*KACS by **3.2**×. Moreover, all of our methods are consistently faster than FastOnline across all datasets. In particular, Opt-Record + P achieves an average speedup of **10.4**× over FastOnline, with the largest performance gaps observed on DBpedia (**24.4**×) and YAGO (**19.4**×), demonstrating the superior efficiency of our framework.

Detailed runtime analysis of our methods. Table 4 presents the runtime breakdown of our proposed methods across four datasets, where the overall cost is divided into the construction time of the G_{SMP} and the remaining steps. The Basic method incurs the highest runtime on all datasets, as the construction of G_{SMP} dominates the total cost due to exhaustive semantic exploration. Opt-BSS significantly reduces the runtime by adopting a binary semantic search strategy, which avoids repeated full graph construction. Building upon this, Opt-Record further improves efficiency by reusing intermediate results during the search process, consistently achieving the lowest runtime among non-parallel methods. Moreover, the parallelized variants (+P) provide substantial additional speedups, especially on large and semantic-rich datasets such as DBpedia, where the cost of G_{SMP} construction is dramatically reduced. Overall, these results demonstrate that the proposed optimization strategies are complementary and jointly enable efficient and scalable r-SICS on large heterogeneous information networks.

6.4 Parameter sensitivity

Effect of $\hat{l}$. Figure 6 shows that the runtime of all methods increases as $\hat{l}$ grows. This is expected, since a larger $\hat{l}$ leads to the detection of more meta-paths, which increases the cost of G_{SMP} construction as well as the subsequent core decomposition process. As shown in Fig. 7, the NMP metric consistently increases with $\hat{l}$, indicating that larger hop limits enable richer semantic diversity by incorporating more meta-path instances. Meanwhile, SC remains stable or exhibits a mild increasing trend across datasets. This suggests that although more connections are introduced when $\hat{l}$ increases, the semantic consistency of the discovered SICs is well preserved. Overall, a larger $\hat{l}$ provides more opportunities to construct semantically coherent communities without sacrificing structural or semantic quality.

Effect of k. Figure 8 shows that the runtime increases as k grows, which is expected since a larger k imposes a stricter k-core constraint. This in turn increases the construction time of G_{SMP} as well as the computational cost of the subsequent core decomposition process. As

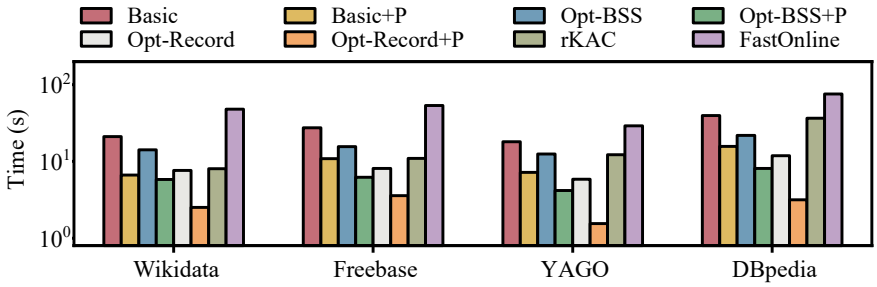

Fig. 5 Efficiency comparison.

Table 4. The runtime (sec) of each step.

Methods	Wiki		Freebase		YAGO		DBpedia	
	G_{SMP}	Other	G_{SMP}	Other	G_{SMP}	Other	G_{SMP}	Other
Basic	22.38	7.462	21.37	6.028	13.88	3.916	28.86	8.162
Opt-BSS	15.05	6.665	9.660	4.278	7.770	3.330	19.60	8.460
Opt-Record	10.44	6.456	4.017	2.483	4.202	2.598	12.36	7.716
Basic+P	4.940	2.210	16.64	5.256	2.964	0.936	4.560	1.708
Opt-BSS+P	3.876	1.824	4.148	1.952	1.904	0.896	3.944	1.856
Opt-Record+P	2.460	1.640	2.880	1.920	0.840	0.560	2.460	1.640

shown in Fig. 9, the NMP metric consistently increases with k, indicating that stronger structural constraints encourage the incorporation of more semantically similar meta-paths, thereby enhancing semantic diversity. Meanwhile, SC remains largely stable across most datasets, with only a mild decrease observed in some cases. This suggests that although larger k may introduce additional connections to satisfy the structural constraint, the semantic consistency of the discovered SICs is generally well preserved.

Effect of r. Figure 10 shows the impact of r on efficiency. As r increases, the runtime naturally grows since more communities need to be enumerated and validated. Figure 11 reports the effect of r on effectiveness metrics. As r increases, the NMP metric consistently rises, indicating that returning more communities introduces richer semantic diversity. Meanwhile, SC exhibits a gradual decreasing trend. This behavior is consistent with the Semantic Monotonicity property defined in Problem 2.3, where communities with lower semantic priority are progressively included. Notably, the decrease in SC remains moderate, suggesting that we can provide multiple community results while largely preserving semantic consistency.

Effect of b. Since b only affects the efficiency of Opt-BSS, Opt-Record, and Opt-Record+P, we only present their corresponding results in Fig. 12. Note that a moderate value of b can improves efficiency by covering more relevant edge types to the semantics S in SMP in one batch. However, an overly large b introduces irrelevant edge types, increasing G_{SMP} construction overhead. Thus, the runtime first reduces and then increases with increasing b.

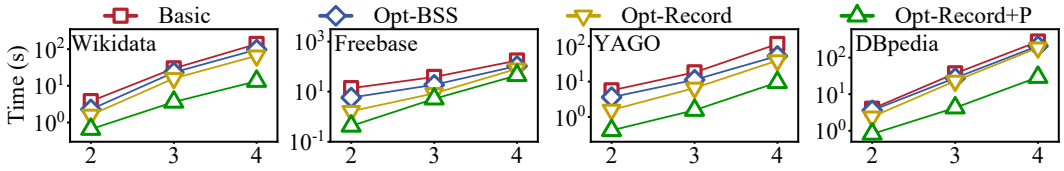

Fig. 6 Effect of $\hat{l}$ on efficiency.

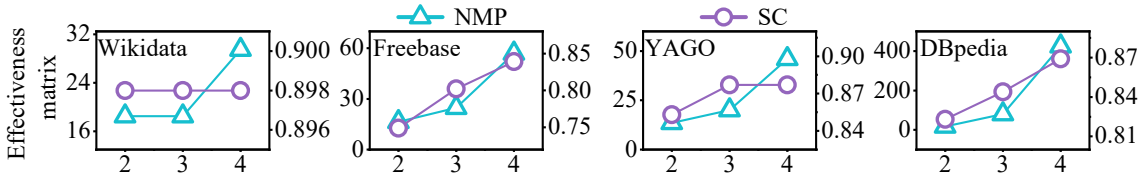

Fig. 7 Effect of $\hat{l}$ on effectiveness metrics (NMP and SC).

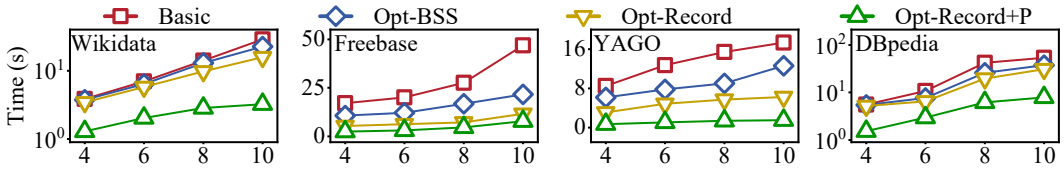

Fig. 8 Effect of k on efficiency.

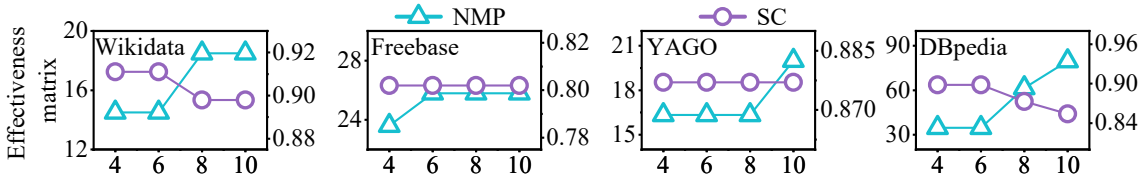

Fig. 9 Effect of k on effectiveness metrics (NMP and SC).

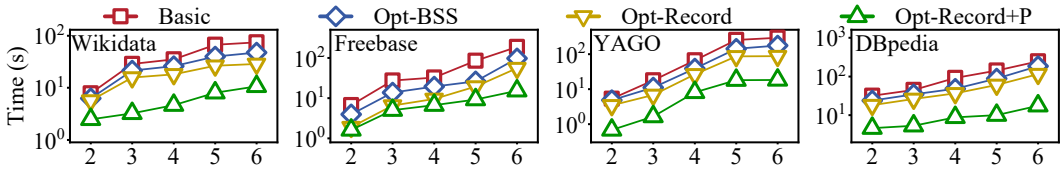

Fig. 10 Effect of r on efficiency.

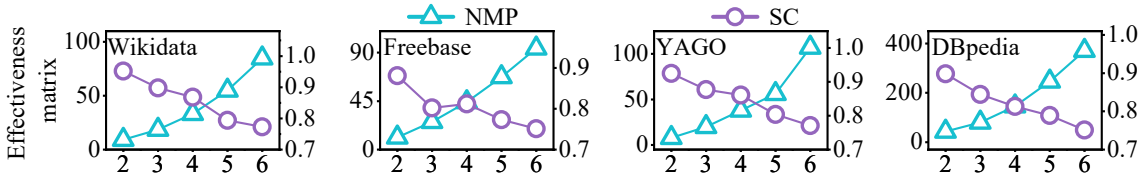

Fig. 11 Effect of r on effectiveness metrics (NMP and SC).

https://doi.org/10.48130/ker-0026-0008

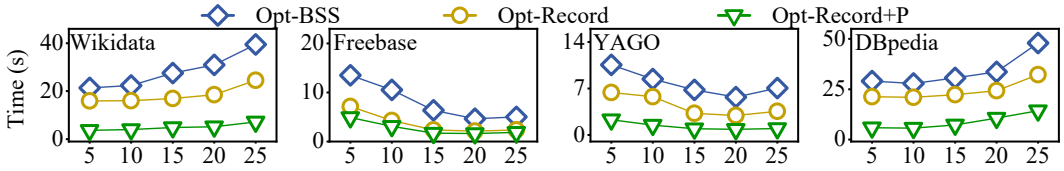

Fig. 12 Effect of $b$on efficiency.

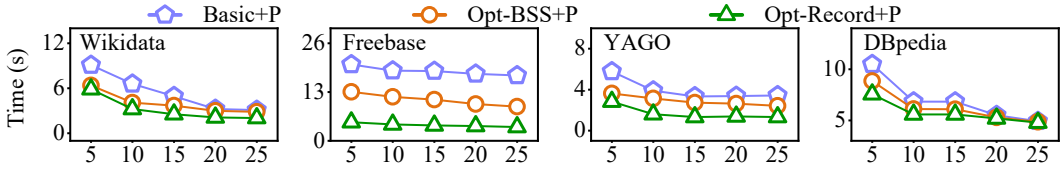

Fig. 13 Effect of # threads on efficiency.

Effect of # threads. We provide the efficiency results of methods using parallel $\mathcal{P}^*$-neighbor expansion in Fig. 13. As # thread increases, the runtime naturally decreases, but beyond a certain point, thread scheduling and communication overhead become dominant factors, causing performance to stabilize.

6.5 Case studies

To qualitatively evaluate the effectiveness and interpretability of our methods, we conduct case studies on DBpedia, as shown in Fig. 14. In both cases, r-SICS consistently discovers communities corresponding to different semantic layers rather than a single semantic view, capturing semantic diversity while yielding communities that are structurally and semantically cohesive and contextually interpretable across different levels of semantic abstraction.

Case study for Ariana Grande. We set the query node as Ariana Grande with k = 8 and and SMP = ⟨Person, Song, Artist, 3⟩, which captures music-related collaboration semantics. The top-3 SICs returned by r-SICS clearly correspond to three distinct semantic layers. The first SIC (red) forms a singer collaboration community, consisting of artists with frequent direct collaborations, such as Nicki Minaj and Selena Gomez. This layer reflects highly specific and strongly cohesive performance-level collaboration semantics. The second SIC (green) evolves into a music production community, incorporating collaborators such as Max Martin and Iggy Azalea. This community captures songwriting- and production-level relations that are semantically close to singer collaborations but structurally different. The third SIC (yellow) further expands into a broader entertainment community, including figures such as Leonardo DiCaprio, reflecting associations between music and entertainment industry. Together, these three SICs demonstrate that r-SICS can effectively uncover layered semantic structures, ranging from artist-level collaborations to entertainment community, while maintaining structural cohesiveness.

Case study for Spielberg. For Steven Spielberg, we set k = 7 and SMP = ⟨Person, Film, Director, 3⟩. The discovered SICs exhibit a clear profession-oriented semantic layering. The most cohesive SIC (orange) corresponds to a core director–editor layer, including long-term collaborators such as Michael Kahn and Frank Morriss, capturing coupled production-level semantics. The second SIC (purple) expands to an actor collaboration layer, incorporating actors such as Christopher Lee and Dianne Kay. The third SIC (green) further forms a film production support layer, including cinematographers and audiovisual experts such as Allen Daviau and William A. Fraker. Across these layers, r-SICS provides multiple semantically coherent communities that reflect different professional roles and collaboration scopes within the film industry.

7 Discussion

7.1 Main findings in relation to existing assumptions

This study was motivated by two core limitations of existing community search methods on HINs, including semantic incompleteness caused by strict reliance on predefined meta-paths and the lack of diversity due to returning a single community. Our results suggest that abstracting semantic intention through SMP can effectively alleviate the inflexibility of predefined meta-path specification. By dynamically exploring semantically relevant relations rather than

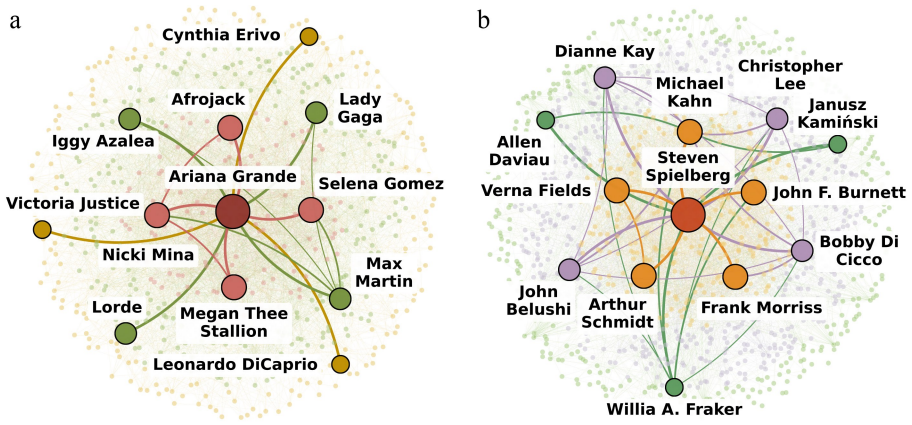

Fig. 14 Case studies on DBpedia.

enumerating fixed patterns, the proposed framework better captures diverse yet semantically coherent communities. More importantly, the output of r-SICS is intended to reflect the kind of answers needed for semantic exploration in semantic-rich HINs. Rather than receiving only one fixed community, users may benefit from a ranked set of communities that begins with the most semantically focused result and then gradually expands to broader but still relevant alternatives. In this sense, returning the top-r SICs is not merely a modeling choice, but also a user-oriented answer format for semantic exploration.

7.2 Methodological advancement

Compared with traditional (k, $\mathcal{P}$)-core based methods, this work advances community search in three aspects. First, at the modeling level, it replaces fixed meta-path specification with semantic-guided exploration via SMP, improving flexibility in semantic-rich HINs. Second, at the problem-definition level, it formalizes semantically important communities (SICs) and organizes community search as a top-r ranked retrieval task. Third, at the algorithmic level, it develops Opt-BSS and Opt-Record to improve search efficiency by avoiding repeated reconstruction and redundant refinement. In addition, the multithreaded expansion introduced in section 5 serves as an implementation-level acceleration technique for large-scale deployment.

7.3 Limitations

Despite its effectiveness, several limitations remain. First, semantic similarity estimation relies on pre-trained heterogeneous graph embeddings, whose quality may be sensitive to noise or sparsity. Second, the current framework assumes static HINs and does not explicitly address dynamic or temporal graph evolution. Third, parameters such as the length bound $\hat{l}$, structural constraint k, and batch size in semantic expansion may influence performance and require manual tuning. Fourth, because no existing baseline exactly matches the r-SICS setting, the empirical comparisons in this paper are conducted under adapted best-effort settings, which may not fully reflect all aspects of advantage or disadvantage.

7.4 Future work

Future research may proceed in several directions. (1) Developing adaptive semantic expansion strategies that incorporate user feedback or learning mechanisms. (2) Extending r-SICS to dynamic heterogeneous graphs with incremental maintenance of semantic subgraphs. (3) Integrating SMP-based community discovery with large language models to support community-level retrieval-augmented generation. (4) Designing more expressive SMP variants that support multiple anchors or composite semantic constraints.

Overall, this study establishes a flexible and scalable paradigm for ranked semantic community discovery in heterogeneous information networks, providing a foundation for further exploration at the intersection of semantic modeling and graph mining.

8 Related work

Community search (CS) is a fundamental problem in graph mining, aiming to identify a cohesive subgraph that contains a given query node. Existing studies can be categorized based on graph homogeneity and how semantics are incorporated.

CS on homogeneous networks. Early CS studies in homogeneous graphs emphasized structural cohesiveness based on connectivity measures such as k-core[26,47], k-truss[29,48], and k-clique[49,50]. These models effectively capture dense substructures but ignore semantic information encoded in edges. Subsequent works incorporated edge weights[51,52] or textual labels[53] into structural metrics to improve interpretability. While these methods enhance interpretability, they fundamentally operate on simple graphs and treat semantics as auxiliary information attached to edges or nodes. As a result, they are not designed to capture complex relational semantics that arise from heterogeneous entity types and relations.

CS on heterogeneous information networks. For HINs, meta-paths $\mathcal{P}$ have become the dominant mechanism for modeling semantic relations across different node and edge types. A series of meta-path-based community search models have been proposed, such as (k,$\mathcal{P}$)-core and (k,$\mathcal{P}$)-truss[1,3,25,54], which extend classical structural cohesiveness by restricting connectivity to instances of a predefined meta-path. These approaches enable users to explicitly encode semantic intent into the community definition and have demonstrated effectiveness on structured HINs. However, relying solely on symmetric meta-paths is often insufficient to preserve semantic consistency within communities, especially in semantic-rich HINs. In such networks, similar relational semantics can be expressed by multiple structurally distinct meta-paths, including both symmetric and asymmetric ones. Restricting community search to a single symmetric meta-path lead to the loss of semantically equivalent but structurally different relational information, thereby affecting the completeness of the resulting communities. These observations highlight the need for more flexible semantic modeling in community search on semantic-rich HINs.

Semantic community search. Beyond the traditional community search paradigm, some recent studies have attempted to incorporate multiple meta-path constraints or result diversity into graph mining tasks[13,15] to alleviate the limitations imposed by a single semantic view. However, these methods are still primarily built upon symmetric meta-paths and consequently lose the approximate semantic information captured by asymmetric meta-paths. In semantic-rich heterogeneous information networks, similar or approximate relational semantics can often be expressed through multiple structurally distinct meta-paths, including both symmetric and asymmetric ones. Expanding community search only among multiple symmetric meta-paths may therefore overlook important semantic relations captured by asymmetric meta-paths, resulting in incomplete community results from a semantic perspective. Moreover, most existing methods still focus on returning a single community and do not explicitly model the fact that a given query node may correspond to multiple semantically plausible communities. In practical applications, users often expect multiple candidate communities that reflect different semantic scopes or levels of abstraction. This practical demand further highlights the importance of jointly supporting flexible semantic modeling and multi-result community search in semantic-rich heterogeneous information networks.

9 Conclusions

We systematically study the top-r semantically important community search (r-SICS) problem on semantic-rich HINs using a novel (k,SMP)-core model based on a flexible and expressive semantic meta-path pattern (SMP). We present a Basic algorithm followed by two optimizations with a semantic-level binary search and an intermediate recording strategy, and finally a parallel expansion to further improve efficiency. Extensive experiments on real-world datasets further confirm the effectiveness and efficiency of our approach in discovering diverse yet semantically coherent communities.

https://doi.org/10.48130/ker-0026-0008

Author contributions

The authors confirm their contributions to the paper as follows: study conception and design: Zhang D, Wang Y, Gu C, Ke X, Xu X; implementation of the proposed method: Zhang D, Wang Y, Gu C; evaluation of the solution: Ke X, Xu X; analysis and interpretation of results: Zhang D, Gu C; draft manuscript preparation: Zhang D, Wang Y; response to reviewers' comments and revision of the results section: Zhang D, Wang Y, Fang Q. All authors reviewed the results and approved the final version of the manuscript.

Data availability

All datasets used in this study are publicly available. The source code and additional materials are available at an anonymized public repository: https://anonymous.4open.science/r/rSICS-8FF5.

Acknowledgments

This work was supported by the National NSF of China (Grant No. 62572162), the Primary R & D Plan of Zhejiang (Grant No. 2023C03198), the 'Pioneer' and 'Leading Goose' R & D Program of Zhejiang (Grant No. 2024C01020).

Conflict of interest

The authors declare that they have no conflict of interest.

Dates

Received 1 March 2026; Revised 2 April 2026; Accepted 13 April 2026; Published online 6 August 2026

References

[1] Fang Y, Yang Y, Zhang W, Lin X, Cao X. 2020. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment* 13(6):854−867

[2] Fang Y, Wang K, Lin X, Zhang W. 2021. Cohesive subgraph search over big heterogeneous information networks: applications, challenges, and solutions. In *Proceedings of the 2021 International Conference on Management of Data. June 20–25, 2021, Virtual Event, China*. New York, USA: ACM. pp. 2829–2838 doi: 10.1145/3448016.345753

[3] Wang Y, Gu C, Xu X, Zeng X, Ke X, et al. 2024. Efficient and effective (k, p)-core-based community search over attributed heterogeneous information networks. *Information Sciences,* 661:120076

[4] Liu Q, Zhu Y, Zhao M, Huang X, Xu J, et al. 2020. Vac: Vertex-centric attributed community search. *2020 IEEE 36th International Conference on Data Engineering (ICDE), april 20–24, 2020. Dallas, TX, USA*. USA: IEEE. pp. 937–948 doi: 10.1109/ICDE48307.2020.00086.

[5] Zhang Z, Huang X, Xu J, Choi B, Shang Z. 2019. Keyword-centric community search. In *2019 IEEE 35th International Conference on Data Engineering (ICDE). April 8–11, 2019. Macao, China*. USA: IEEE. pp. 422–433 doi: 10.1109/ICDE.2019.00045.

[6] Chen L, Liu C, Zhou R, Li J, Yang X, et al. 2018. Maximum co-located community search in large scale social networks. *Proceedings of the VLDB Endowment* 11(10):1233−1246

[7] Zhang F, Zhang Y, Qin L, Zhang W, Lin X. 2017. Finding critical users for social network engagement: the collapsed k-core problem. *Proceedings of the AAAI Conference on Artificial Intelligence* 31(1):245−251

[8] Ley M. 2002. The dblp computer science bibliography: evolution, research issues, perspectives. In *String Processing and Information Retrieval*. Berlin, Heidelberg: Springer. pp. 1–10 doi: 10.1007/3-540-45735-6_1

[9] Tang J. 2016. Aminer: Toward understanding big scholar data. *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. San Francisco California USA* . New York, USA: ACM. pp. 467 doi: 10.1145/2835776.2835849

[10] Mendes P, Jakob M, Bizer C. 2012. Dbpedia: a multilingual cross-domain knowledge base. *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012). 21−27 May 2012, Istanbul, Turkey*. European Language Resources Association (ELRA). pp. 1813−1817 doi: 10.63317/4b94v6njihis

[11] Hoffart J, Suchanek FM, Berberich K, Weikum G. 2013. Klaus Berberich, and Gerhard Weikum. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence* 194:28−61

[12] Yao K, Chang L . 2021. Efficient size-bounded community search over large networks. *Proceedings of the VLDB Endowment* 14(8):1441−1453

[13] Ye J, Zhu Y, Chen L. 2023. Top-*r* keyword-based community search in attributed graphs. *2023 IEEE 39th International Conference on Data Engineering (ICDE). April 3–7, 2023, Anaheim, CA, USA*. USA: IEEE. pp. 1652–1664 doi:10.1109/ICDE55515.2023.00130.

[14] Wang Y, Gou X, Xu X, Geng Y, Ke X, et al. 2024. Scalable community search over large-scale graphs based on graph transformer. *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 14–18 July 2024, Washington D.C., USA*. New York, USA: ACM. pp. 1680–1690 doi: 10.1145/3626772.3657771

[15] Jiang Y, Fang Y, Ma C, Cao X, Li C . 2022. Effective community search over large star-schema heterogeneous information networks. *Proceedings of the VLDB Endowment* 15(11):2307−2320

[16] Xu X, Liu J, Wang Y, Ke X. 2022. Academic expert finding via (k,$\mathcal{P}$)-core based embedding over heterogeneous graphs. *2022 IEEE 38th International Conference on Data Engineering (ICDE). 9–12 May 2022, Kuala Lumpur, Malaysia*. USA: IEEE. pp. 338–351doi: 10.1109/icde53745.2022.00030

[17] Sozio M, Gionis A. 2010. The community-search problem and how to plan a successful cocktail party. *KDD'10: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 25–28, 2010, Washington D.C., USA*. New York, NY, USA: Association for Computing Machinery. pp. 939–948 doi: 10.1145/1835804.1835923

[18] Wang Y, Liu J, Xu X, Ke X, Wu T, et al. 2023. Efficient and effective academic expert finding on heterogeneous graphs through (k,$\mathcal{P}$)-core based embedding. *ACM Transactions on Knowledge Discovery from Data* 17(6):1−35

[19] Dudley JT, Deshpande T, Butte AJ. 2011. Exploiting drug-disease relationships for computational drug repositioning. *Briefings in Bioinformatics* 12(4):303−311

[20] Pesántez-Cabrera P, Kalyanaraman A. 2019. Efficient detection of communities in biological bipartite networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 16(1):258−271

[21] Guo Z, Xia L, Yu Y, Ao T, Huang C. 2024. LightRAG: simple and fast retrieval-augmented generation. *arXiv* Preprint

[22] Edge D, Trinh H, Cheng N, Bradley J, Chao A, et al. 2024. From local to global: a graph rag approach to query-focused summarization. *arXiv* Preprint

[23] Wang S, Fang Y, Zhou Y, Liu X, Ma Y. 2026. ArchRAG: attributed community-based hierarchical retrieval-augmented generation. *Proceedings of the AAAI Conference on Artificial Intelligence* 40(19):15868−15876

[24] YLiu Y, Guo F, Xu B, Bao P, Shen H, et al. 2023. Significant-attributed community search in heterogeneous information networks. *arXiv* Preprint

[25] Yang Y, Fang Y, Lin X, Zhang W. 2020. Effective and efficient truss computation over large heterogeneous information networks. *2020 IEEE 36th International Conference on Data Engineering (ICDE). April 20–24, 2020. Dallas, TX, USA* . USA: IEEE. pp. 901–912 doi: 10.1109/icde48307.2020.00083

[26] Barbieri N, Bonchi F, Galimberti E, Gullo F. 2015. 2015. Efficient and effective community search. *Data Mining and Knowledge Discovery* 29(5):1406−1433

[27] Cui W, Xiao Y, Wang H, Wang W. 2014. Local search of communities in large graphs. *SIGMOD '14: Proceedings of the 2014 ACM SIGMOD*

International Conference on Management of Data, June 22–27, 2014, Snowbird, Utah, USA. New York, NY, USA: Association for Computing Machinery. pp. 991–1002 doi: 10.1145/2588555.2612179

[28] Huang X, Cheng H, Qin L, Tian W, Yu JX. 2014. Querying k-truss community in large and dynamic graphs. *SIGMOD '14: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, June 22–27, 2014, Snowbird, Utah, USA*. New York, NY, USA: Association for Computing Machinery. pp. 1311–1322 doi: 10.1145/2588555.2610495

[29] Huang X, Lakshmanan LVS, Yu JX, Cheng H . 2015. Approximate Closest Community Search in Networks. *PVLDB* 9(4):276−287

[30] Wang Y, Khan A, Wu T, Jin J, Yan H. 2020. Semantic guided and response times bounded top-k similarity search over knowledge graphs. *2020 IEEE 36th International Conference on Data Engineering (ICDE), April 2020, Dallas, TX, USA*. USA: IEEE. pp. 445–456 doi: 10.1109/icde48307.2020.00045

[31] Chen L, Liu C, Liao K, Li J, Zhou R. 2019. Contextual community search over large social networks. *2019 IEEE 35th International Conference on Data Engineering (ICDE), April 8-11, 2019. Macao, China*. USA: IEEE. pp. 88–99 doi: 10.1109/ICDE.2019.00017

[32] Campana P, Varese F. 2022. Studying organized crime networks: data sources, boundaries and the limits of structural measures. *Social Networks* 69:149−159

[33] Chattoe E, Hamill H. 2005. It's not who you know—it's what you know about people you don't know that counts: extending the analysis of crime groups as social networks. *The British Journal of Criminology* 45(6):860−876

[34] Wang T, Rudin C, Wagner D, Sevieri R. 2013. Learning to detect patterns of crime. In *Machine Learning and Knowledge Discovery in Databases*, eds. Blockeel H, Kersting K, Nijssen S, Železný F. Berlin, Heidelberg: Springer. pp. 515–530 doi: 10.1007/978-3-642-40994-3_33

[35] Wang Y, Khan A, Xu X, Jin J, Hong Q, et al. 2022. Aggregate queries on knowledge graphs: Fast approximation with semantic-aware sampling. *2022 IEEE 38th International Conference on Data Engineering (ICDE). May 9–12, 2022, Kuala Lumpur, Malaysia*. USA: IEEE. pp. 2914–2927 doi: 10.1109/icde53745.2022.00263

[36] Zhang Z, Huang X, Xu J, Choi B, Shang Z. 2019. Keyword-centric community search. *2019 IEEE 35th International Conference on Data Engineering (ICDE). April 8–11, 2019 Macao, China*. USA: IEEE. pp. 422–433 doi: 10.1109/ICDE.2019.00045

[37] Sun Z, Deng ZH, Nie JY, Tang J. 2019. Rotate: knowledge graph embedding by relational rotation in complex space. *arXiv* Preprint

[38] Huang X, Zhang J, Li D, Li P. 2019. Knowledge graph embedding based question answering. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM '19, New York, NY, USA, 2019*. USA: Association for Computing Machinery. pp. 105–113 doi: 10.1145/3289600.3290956

[39] Batagelj V, Zaveršnik M. 2003. An O(m) algorithm for cores decomposition of networks. *arXiv* Preprint

[40] Anonymous GitHub. 2025. *Top-r semantically important community search on semantic-rich heterogeneous graphs (r-SICS)*. https://anonymous.4open.science/r/rSICS-42B6

[41] Vrandečić D, Krötzsch M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78−85

[42] Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, 2008, New York, NY, USA*. USA: Association for Computing Machinery. pp. 1247–1250 doi: 10.1145/1376616.1376746

[43] Rebele T, Suchanek F, Hoffart J, Biega J, Kuzey E, et al. 2016. YAGO: a multilingual knowledge base from wikipedia, wordnet, and geonames. In *The Semantic Web – ISWC 2016*, eds. Groth P, Simperl E, Gray A, Sabou M, Krötzsch M, et al. Cham: Springer. pages 177–185 doi: 10.1007/978-3-319-46547-0_19

[44] Sun Y, Han J, Yan X, Yu PS, Wu T . 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4(11):992−1003

[45] Meng C, Cheng R, Maniu S, Senellart P, Zhang W. 2015. Discovering meta-paths in large heterogeneous information networks. *WWW '15: Proceedings of the 24th International Conference on World Wide Web, May 18–22, 2015, Florence, Italy*. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee. pp. 754–764 doi: 10.1145/2736277.2741123

[46] Shi C, Li Y, Zhang J, Sun Y, Yu PS. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29(1):17−37

[47] Fang Y, Wang Z, Cheng R, Wang H, Hu J. 2019. Effective and efficient community search over large directed graphs. *IEEE Transactions on Knowledge and Data Engineering* 31(11):2093−2107

[48] Liu Q, Zhao M, Huang X, Xu J, Gao Y. 2020. Truss-based community search over large directed graphs. *SIGMOD '20: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery. pp. 2183–2197 doi: 10.1145/3318464.3380587

[49] Cui W, Xiao Y, Wang H, Lu Y, Wang W. 2013. Online search of overlapping communities. *SIGMOD '13: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, June 22–27, 2013, New York, USA*. New York, NY, USA: Association for Computing Machinery. pp. 277–288 doi: 10.1145/2463676.2463722

[50] Yuan L, Qin L, Zhang W, Chang L, Yang J . 2018. Index-based densest clique percolation community search in networks. *IEEE Transactions on Knowledge and Data Engineering* 30(5):922−935

[51] Zheng D, Liu J, Li RH, Aslay Ç, Chen YC, et al. 2017. Querying intimate-core groups in weighted graphs. *Proceedings of the 2017 IEEE 11th International Conference on Semantic Computing (ICSC), San Diego, CA, USA, 30 January – 1 February, 2017*. Piscataway, NJ, USA: IEEE. pp. 156–163 doi: 10.1109/ICSC.2017.80

[52] Li L, Zhao Y, Luo S, Wang G, Wang Z. 2023. Efficient community search in edge-attributed graphs. *IEEE Transactions on Knowledge and Data Engineering* 35(10):10790−10806

[53] Habib WMA, Mokhtar HMO, El-Sharkawi ME. 2022. Discovering top-weighted k-truss communities in large graphs. *Journal of Big Data* 9(1):36

[54] Zhou Y, Fang Y, Luo W, Ye Y. 2023. Influential community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment* 16(8):2047−2060