Effective and Efficient Community Search over Large-Scale Hypergraphs

Yu Liu
Shandong University
Qingdao, China
yuliu@mail.sdu.edu.cn

Qi Luo*
University of New South Wales
Sydney, Australia
qi.luo1@unsw.edu.au

Yanwei Zheng
Shandong University
Qingdao, China
zhengyw@sdu.edu.cn

Wenjie Zhang
University of New South Wales
Sydney, Australia
wenjie.zhang@unsw.edu.au

Xuemin Lin
Shanghai Jiao Tong University
Shanghai, China
xuemin.lin@sjtu.edu.cn

Dongxiao Yu
Shandong University
Qingdao, China
dxyu@sdu.edu.cn

ABSTRACT

Community search (CS) is a fundamental task in graph mining with a wide range of applications, such as social network, recommendation and link prediction. Traditional CS approaches fail to capture the higher-order interactions in hypergraphs, where vertex-centric community models fail to capture the relationships formed by hyperedges involving multiple vertices. This leads to traditional community models in hypergraphs that tend to overlook the overlapness of hyperedges and be overly large. To tackle these issues, we study the problem of Hypergraph Minimum Community Search (HMCS), aiming to identify the minimum cohesive communities in hypergraphs. We propose a new cohesive community model, (k, s)-HCore, where k denotes the level of participation in hyperedge interactions, and s represents the intensity of these interactions. This model effectively captures higher-order interactions in hypergraphs and addresses hyperedge overlapping issues in hypergraphs. Moreover, to solve the issue of overly large communities, we propose a branch-and-bound method (BAB) to search a minimal community for given queries. To further improve computational efficiency, we propose an optimized algorithm (OBBAB) using candidate set pruning, lower bound pruning, and an enhanced branching strategy. Experimental results show that the communities searched by the (k, s)-HCore model are twice as overlapness as traditional models, while reducing community sizes by approximately 25%, demonstrating the effectiveness in capturing minimal communities in hypergraphs. The OBBAB achieves a 10^2 to 10^4 efficiency improvement over BAB, and exhibits linear scalability with increasing dataset sizes. Case studies demonstrate the broad applicability and flexibility of the (k, s)-HCore model across diverse real-world scenarios.

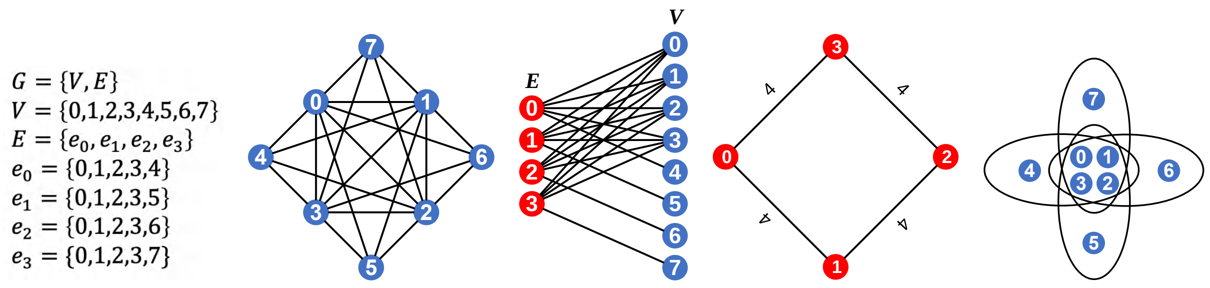
1 INTRODUCTION

Community search (CS)[12, 35] is a common and critical task widely used in complex networks analysis. The goal of CS is to identify communities or groups of closely connected vertices within networks [18], which can be applied in social network [53], recommendation systems [55], link prediction [32]. In recent years, hypergraphs [10, 17, 38], which connect multiple vertices through hyperedges, have become a crucial tool for modeling higher-order interactions and capturing complex group dynamics. The problem of CS in hypergraphs facilitates the discovery of cohesive sub-hypergraphs in higher-order interactions. For instance, Figure 1 illustrates the DBLP academic network modeled as a hypergraph, where co-authors are treated as a cohesive group, highlighting the strength of academic collaborations.

Despite the significant advantages of CS in hypergraphs, there are two major challenges that are often raised when performing CS on hypergraphs. (1) ***Ignoring Hyperedge Overlapness:*** Traditional pairwise cohesive models in CS rely on vertex-centric, including k-core [14], d-core [34], k-edge [11], and k-truss [27]. These models fail to consider overlapness and capture crucial high-order interactions involving multiple vertices. Directly converting hypergraphs to pairwise graphs to match these models will lead to a sharp increase in the number of edges, and result in excessive resource overhead. While the strength of group interaction overlapness is crucial for accurately representing community cohesiveness in hypergraphs, current hypergraph-specific models, such as k-hypercore [42], *Nbr*-k-core [5], and *CoCore* [44], primarily focus on individual vertex connections and ignore group-level interaction strength, making them insufficient for representing cohesive community structures in hypergraphs. As a result, there is an urgent need for a new cohesive model that captures both high-order interactions and group interaction strength. (2) ***Excessive Community Size:*** Existing hypergraph CS methods [16, 47] often face the challenge of producing excessively large communities, due to a lack of emphasis on size minimization. While these methods are effective at identifying cohesive structures, the resulting communities are often too large, causing inefficiencies and making them unsuitable for scenarios requiring smaller, more precise communities. This issue becomes particularly evident in applications such as publication networks [26, 32, 52], influence spreading [8, 13], and hypergraph classification [51, 56], where identifying minimal community structures can significantly improve performance and interpretability. Detailed explanations can be found in Section 3.3 and 6.8. However, identifying such minimal communities is inherently challenging, as the problem is NP-Hard [50], making it computationally infeasible to solve optimally within polynomial time.

To address the aforementioned challenges, we propose the approach called ***H**yperedge-**C**entric **M**inimum **C**ommunity **S**earch (HCMCS)*. This approach ensures that the identified communities are minimal and effectively capture the hyperedge overlapness, overcoming the limitations of traditional vertex-centric models in hypergraphs. Specifically, the HCMCS approach consists of two important components: the (k, s)-HCore model and

*Qi Luo is the corresponding author.



(a) The DBLP network (b) The pairwise graph (c) The bipartite graph (d) The s-line graph (e) The hypergraph

Figure 1: Different representations of a DBLP network. (a) illustrates a hypergraph representation where vertices represent authors and hyperedges represent group interactions (publications). (b) shows the pairwise graph representation, simplifying these interactions into pairwise connections. (c) depicts a bipartite graph, with separate vertex sets for authors and publications. (d) presents the s-line graph, emphasizing relationships between hyperedges based on shared vertices. (e) visualizes a hypergraph preserving the richness of higher-order group interactions. However, the traditional k-core model [42] identifies this hypergraph only as a 1-core, failing to effectively capture cohesive group relationships.

the corresponding minimum community search algorithm, which tackle these difficulties collaboratively.

For challenge 1: Hyperedge-centric interaction modeling. We first propose a novel hyperedge-based (k, s)-HCore model, where k represents the hyperedge connection constraint, and s denotes the hyperedge overlapping constraint. This model operates directly on hypergraphs, avoiding information loss and computational overhead associated with converting hypergraphs to pairwise graphs, which makes the communities more accurate and effective. We propose two key metrics to quantify the complex relationships in hyperedges: *Interaction Engagement (IE)* and *Intersection Strength (IS)*, which correspond to parameters k and s in the (k, s)-HCore.
The IE metric is designed to measure the strength of overlap between hyperedges. It captures richer group interaction information by calculating the frequency of interactions between hyperedges in the hypergraph. By setting a threshold, we can identify strong interactions between hyperedges when their engagement frequency surpasses this threshold. The IE metric not only captures vertex-to-vertex connections but also reflects the cohesion of group-level interactions, providing a more comprehensive and profound understanding of community structures.
The IS metric, on the other hand, is crucial for describing group-level interaction strength, which is an essential factor in determining community cohesiveness within hypergraphs. Traditional methods often overlook this aspect, focusing only on individual vertex connections. However, considering group interaction strength can provide deeper insights into the internal cohesiveness of communities. For this purpose, we propose the IS metric, which uses the s-walk algorithm [3, 39, 41] to calculate the interaction strength between hyperedges by capturing high-order interactions within hypergraphs, effectively reflecting their intensity. Specifically, the IS metric measures the number of shared vertices between hyperedges, representing their intersection strength. By leveraging this approach, we can more accurately describe the internal cohesiveness of communities and perform CS based on intersection strength, thereby identifying highly interactive communities.
By combining the IE and IS metrics, we propose Restricted Interaction Engagement (RIE), a unified framework that captures both hyperedge overlap and interaction strength. RIE addresses the limitations of traditional methods by capturing higher-order interactions, addressing hyperedge overlap issues, and enabling a direct analysis of group-level interactions. This integration allows for more accurate and cohesive community detection, enhancing the (k, s)-HCore model's ability to identify highly interactive communities. RIE not only provides deeper insights into community structures but also supports efficient algorithmic pruning and search strategies, making it highly effective for large-scale hypergraph applications like influence spreading and classification.

For challenge 2: Minimum community search. Considering the excessive community size, we utilize a basic branch-and-bound (BAB) framework to implement a minimum community search by systematically exploring all possible branches, checking each community to determine if it meets the minimum (k, s)-HCore conditions, and then eliminating those that do not. However, BAB often explores multiple branches, many of which are unproductive, leading to inefficiencies. To overcome this limitation, we propose an enhanced OBBAB algorithm, integrating candidate set pruning, lower bound pruning, and improved branching strategy. These optimizations aim to reduce computational complexity and enhance the efficiency of identifying minimal communities, thereby making the solution more practical for real-world applications. Furthermore, we have rigorously validated the efficacy and correctness of these pruning methods through extensive proofs.

We summarize our contributions as follows:

- **Innovative Cohesive Model.** This paper is pioneering in addressing the hypergraph community search problem with a focus on hyperedge interactions. To accurately capture these interactions, we introduce the concepts of interaction engagement, which limits the frequency of interactions, and interaction strength, which limits the degree of interactions. We integrate these concepts to propose a hyperedge-centric cohesive model, (k, s)-HCore. This model addresses the challenges of varying hyperedge sizes and the complexity of hyperedge intersections, thereby improving the precision of community searches within hypergraphs. Additionally, we introduce the Hyperedge-Centric Minimum Community Search problem, based on (k, s)-HCore, to further refine hypergraph community searches.

- **Effective and Efficient Algorithms.** We have proven that the minimum community search problem is NP-hard. To address this, we develop a branch-and-bound algorithm enhanced with three optimal techniques: candidate set pruning, lower bound pruning, and a strategic branching method. These techniques collectively improve the computational efficiency of our algorithm.
- **Comprehensive Experiments.** We conduct extensive experiments on eight datasets to validate the model's effectiveness and algorithm's efficiency. The (k, s)-HCore model achieves higher k values compared to traditional vertex-centric models. The overlapness metric of (k, s)-HCore is twice that of traditional k-core and *CoCore*, while community sizes are reduced by 25%. Our optimization strategies improve algorithm efficiency by 2-4 orders of magnitude. Metrics such as community density and triple count demonstrate a 10%-30% improvement in cohesion over traditional models. Case studies further highlight the superiority of (k, s)-HCore in large-scale hypergraphs.

2 RELATION WORK

We introduce the work related to the cohesive subgraph and the community search, respectively.

Cohesive Subgraph. The mining of cohesive subgraphs is an important study area of graph analysis. The concept of k-core is first proposed in graphs by [7, 48] and introduced in hypergraph by [42, 46]. Furthermore, Malliaros et al. [45] suggested a link between k-core and community engagement, which makes it better to be used for CS. Huang et al. [29] proposed the concept of k-truss, which is further restricted from the perspective of edges. To find different regions containing different densities in graphs, Govindan et al. [25] proposed the k-peak. The concept of k-ECC has been introduced in [11], which is defined as the property of a graph to remain connected even after removing $k-1$ edges. Gabert et al. [23] proposed the concept of the nucleus, redefines cohesive subgraphs, and unifies the concepts of k-core, and k-truss. However, compared to pairwise graphs, there is relatively less research on the concepts and algorithms for cohesive subgraphs in hypergraphs. Luo et al. [42, 43] introduced a concept called Hypercore in hypergraphs, which corresponds to the k-core in pairwise graphs. Arafat et al. [5] have proposed *Nbr-k*-core, a methodology to depict closely cohesive subgraphs from the perspective of vertex neighbors. Luo et al. [44] introduced the *CoCore* considering both group engagement and neighbor engagement.

Community Search. The purpose of the CS is to find a densely connected subgraph in graphs based on the given queries [28]. The most classical method for CS is to utilize the cohesiveness models [1, 19, 21, 36, 40, 54, 57, 58]. This approach involves analyzing the connections and interactions among members within a community to identify groups of members that are closely connected to some degree, such as [2, 14, 27, 50]. However, researchers have found that if the size of the community is not limited, the resulting communities can be excessively large, making it difficult to obtain the desired results intuitively. To this end, Barbieri et al. [6] proposed a minimum CS problem, which involves finding a community that satisfies the k-core condition and has the minimum number of vertices and Dong et al. [15] proposed the search for a minimum butterfly core community. Additionally, researchers have started to expand the CS to encompass more types of graphs beyond simple graphs. They have begun to conduct CS on more complex graph structures, such as directed graphs [20, 24], heterogeneous graphs [22], attribute graphs [37], and temporal graph [33]. Nevertheless, there is no minimum CS specifically designed for hypergraphs currently.

In this paper, we propose the (k, s)-HCore model, focusing on hyperedge interactions to identify cohesive sub-hypergraphs. Unlike previous vertex-centric models, which are prone to the influence of individual vertices, the (k, s)-HCore leverages the flexibility of hyperedges to overcome these limitations. Based on this model, we address the *HCMCS* problem, enabling the discovery and search of minimal communities in hypergraphs.

3 PRELIMINARIES AND APPLICATION

3.1 Notation Definition

Let $G = (V, E)$ represent an undirected and unweighted hypergraph with V and E denoting the sets of vertices and hyperedges, respectively. Consider $S \subseteq G$ to be a sub-hypergraph induced by hyperedges. Unless otherwise specified, the sub-hypergraph refers to a hyperedge-induced sub-hypergraph. We denote $|S|$ as the number of hyperedges in S. Furthermore, let $NV_G(u)$ signify the set of hyperedges incident to a vertex u, defined as $NV_G(u) = \{e \mid u \in e, \forall e \in G\}$, and let $NE_G(e)$ represent the set of hyperedges incident to a hyperedge e, defined as $NE_G(e) = \{e' \mid e \cap e' \neq \emptyset, e' \in G\}$. When the context is clear, we omit the footnote of symbols, such as using $NV(u)$ instead of $NV_G(u)$. We first propose the concept of interaction engagement, which measures the frequency of interaction between a hyperedge and other hyperedges.

Definition 1. *(**I**nteraction **E**ngagement (IE)) Given a hypergraph $G = (V, E)$, let $e \in E$ be an arbitrary hyperedge. The interaction engagement of e is defined as the number of hyperedges that intersect with e, which is computed by the following formula:*

$$IE_G(e) = |\{e' \in NE(e) | e' \cap e \neq \emptyset\}|. \tag{1}$$

To quantitatively ascertain the degree of interaction between two hyperedges in a hypergraph, we define the interaction strength as follows.

Definition 2. *(**I**nteraction **S**trength (IS)) Given a hypergraph $G = (V, E)$, where for any two hyperedges e_0 and e_1, their interaction strength is defined as the intersection size between them, which is computed by the following formula:*

$$IS_G(e_0, e_1) = |\{u | u \in e_0 \cap e_1\}|. \tag{2}$$

Based on Definition 1 and 2, we propose a measure called restricted interaction engagement, which quantifies the frequency of hyperedge interaction under the constraint of interaction strength.

Definition 3. *(**R**estricted **I**nteraction **E**ngagement (RIE)) Given a hypergraph $G = (V, E)$, let $e \in E$ be an arbitrary hyperedge. The restricted interaction engagement of e is defined as the number of hyperedges that intersect with e, and their interaction strength is not less than the threshold s, which is computed by the following formula:*

$$RIE_G(s, e) = |\{e' \in IE(e) | IS(e', e) \geq s\}|. \tag{3}$$

Subsequently, we propose the definition of (k, s)-HCore and hyperedge core number.

Algorithm 1 (k, s)-HCore Decomposition

```
Input: Hypergraph $G = (V, E)$, IS $s$
Output: Core number $cE(\cdot)$
1: $S \leftarrow G, k \leftarrow 1$;
2: while $S$ is not empty do
3:     $e \leftarrow \arg_{e \in S} \min RIE_S(s, e)$;
4:     $k \leftarrow \max(k, RIE_S(s, e))$;
5:     $S \leftarrow S \setminus e$;
6:     $cE_G(s, e) \leftarrow k$;
7: return $cE(\cdot)$;
```

Definition 4. *((k, s)**-HCore**) Given a hypergraph $G = (V, E)$, a hyperedge-induced maximal sub-hypergraph S is a (k, s)-HCore if and only if for any hyperedge $e \in S$, $RIE_S(s, e) \geq k$.*

Definition 5. *(**Hyperedge Core Number**) Given a hypergraph $G = (V, E)$ and the IS s, for any hyperedge $e \in E$, its hyperedge core number $cE_G(s, e)$ is defined as the maximum value of k such that hyperedge e belongs to the (k, s)-HCore.*

Example 1. *For example, in Figure 1, $IE(e_0) = 3$ because it intersects with three hyperedges. $IS(e_0, e_1) = 4$ because these two hyperedges share four common vertices. $RIE(4, e_0) = 3$ because e_0 intersects with e_1, e_2, and e_3, each sharing four common vertices with e_0. The entire graph forms a $(3, 4)$-HCore since the $RIE(4, e)$ for every hyperedge in the graph is equal to 3.*

Based on the above definitions, we present the decomposition algorithm in Algorithm 1 for computing all the hyperedge core numbers in hypergraph $G = (V, E)$ with *IS* s. It entails an iterative removal of hyperedges in E with *RIE* below the prescribed threshold value k. This iterative process continues until all remaining hyperedges in the hypergraph meet the criterion of *RIE* greater than or equal to k. The remaining hypergraph resulting from this iterative process is identified as the (k, s)-HCore sub-hypergraph of the initial hypergraph.

3.2 Problem Definition

In this paper, we focus on the problem of ***Hyperedge-Centric Minimum Community Search (HCMCS)***.

Problem 1. *(**HCMCS**) Given a hypergraph $G = (V, E)$, a query vertex q, positive integers IE k and IS s, the HCMCS problem aims to find a minimum sub-hypergraph $S \subseteq G$ which holds the following properties:*

*(1) **Connectivity Constraint.** S is connected and contains q;*
*(2) **Structure Constraint.** $\forall e \in S$, $RIE_S(s, e) \geq k$;*
*(3) **Size Constraint.** There does not exist a sub-hypergraph S' that satisfies conditions 1), 2), and $|S'| \leq |S|$.*

3.3 Discussion of Application

Hyperedge-Centric Minimum Community Search plays a critically important role in multiple domains in the real world. The follows are descriptions of several practical applications for hypergraph community search, and all applications are verified in experiments.

Publication Networks [26, 32, 52]: In publication networks, researchers from the same university or closely interacting colleagues often collaborate in publications, forming a group (s) that produces a large number (k) of publications. (k, s)-HCore with larger s and k can more easily identify these groups in the publication network. This is advantageous for publishers, as hypergraph-based minimal CS can identify small but tightly connected groups with high publication output. In our experiment 6.8, we conducted a detailed analysis of the historical publication network CoMH, demonstrating the effectiveness of this approach.

Influence Spreading [8, 13]: In social networks and information dissemination, certain key vertices have higher influence and can affect a wide range of the spreading process. By using the (k, s)-HCore model, we can identify these vertices with high centrality and strong influence. Vertices in (k, s)-HCores with high k and low s tend to have greater influence during the spreading process. In our experiment 6.8, we evaluated influence spreading using the SIR diffusion model, and the results showed that the (k, s)-HCore model has significant advantages in selecting influential vertices.

Hypergraph Classification [51, 56]: In data mining and network analysis, hypergraphs from different domains often exhibit similar structural characteristics, such as the distribution of (k, s)-HCores and the range of s values. By analyzing these features, hypergraphs can be effectively classified, improving the accuracy and efficiency of classification. In our experiment 6.8, we demonstrated the similarity in core distribution of datasets from the same domain under (k, s) core constraints, proving the potential of the (k, s)-HCore model in hypergraph classification tasks.

4 NAIVE APPROACH

In this section, we present a baseline (traditional) algorithm, outlined in Algorithm 2, designed to solve the *HCMCS* problem, which seeks the optimal solution by exhaustively enumerating all potential communities that fulfill the *HCMCS* criteria. Considering the NP-hard nature of the problem, the algorithm has been designed with rudimentary pruning strategies to ensure that the minimum community meeting the requirements can be identified within a reasonable timeframe.

Algorithm 2 BAB: BranchAndBound

```
Input: Hypergraph $G = (V, E)$, query vertex $q$, positive integers IE $k$,
       IS $s$
Output: Sub-hypergraph $S$
1: $cE(\cdot) \leftarrow (k, s)$-HCore Decomposition;
2: $G \leftarrow G \setminus \{e | cE(e) < k\}$;        ▷ Reduce the size of the hypergraph
3: if $G$ is empty then                  ▷ No community meeting the criteria
4:     return $\emptyset$;
5: $S \leftarrow G$;                               ▷ Optimal result initialization
6: for $e \in NV_G(q)$ do
7:     $P \leftarrow E(G)$;                        ▷ Candidate set initialization
8:     BranchAndBound($\{e\}, P \setminus \{e\}$);
9: return $S$;
10: procedure BranchAndBound($C, P$)
11:     if $C$ is $(k, s)$-HCore and $|C| <= |S|$ then
12:         $S \leftarrow C$;
13:     else
14:         $e \leftarrow$ Randomly select a hyperedge from P;
15:         BranchAndBound($C \cup \{e\}, P \setminus \{e\}$);
16:         BranchAndBound($C, P \setminus \{e\}$);
```

Algorithm 2 initially performs a (k, s)-HCore decomposition on the hypergraph G to determine the core number of each hyperedge by Algorithm 1. It then prunes hyperedges that do not meet the k threshold, effectively reducing the size of the hypergraph (Lines 1-2). If the hypergraph is empty after pruning, the algorithm 2 concludes with an empty set, indicating that no community satisfies the criteria (Line 3-4). Conversely, the entire G is taken as the initial optimal result S (Line 5).

Subsequently, Algorithm 2 traverses the adjacent hyperedge set $NV_G(q)$ of the query vertex q, performing branch-and-bound methodology for each hyperedge e (Line 6-16). In this process, this initiates with the establishment of the candidate set P as the hyperedge set of the hypergraph G, and iteratively, the algorithm refines the optimal sub-hypergraph via the *BrandAndBound* procedure (Line 10-16). When C satisfies the *HCMCS* criteria and the size of it does not exceed that of the minimum sub-hypergraph S, it is adopted as the new optimal solution (Lines 11-12). Otherwise, Algorithm 2 randomly selects a hyperedge e from P and explores sub-hypergraphs both including and excluding e, iteratively contracting the search domain (Line 14-16).

Through the detailed branch-and-bound process, Algorithm 2 effectively explores the solution space. The recursive application of the *BranchAndBound* procedure ensures that each sequential solution is not only locally optimal within its respective search trajectory, but also that it holds the global optimum.

Correctness Analysis. Algorithm 2 guarantees correctness by exhaustively exploring all sub-hypergraphs that satisfy the *HCMCS* criteria. The branch-and-bound process ensures no valid solutions are omitted, and the smallest community is always identified as the optimal result.

Performance Analysis. Algorithm 2 employs a branch-and-bound method, an NP-Hard algorithm with a time complexity of 2^m, where m represents the number of hyperedges in the hypergraph.

To reduce m, we initially apply the (k, s)-HCore decomposition method to narrow down the search space. This step effectively avoids unnecessary searches by identifying and excluding vertices and hyperedges that are unlikely to belong to any community that meets *HCMCS* criteria (Line 1-2). Furthermore, during the recursive process of the algorithm, boundary conditions are set for inspection (Line 11-12), halting further in-depth search if the current community satisfies the *HCMCS* condition and does not exceed the size of the existing optimal solution. These strategies significantly reduce the number of branches to explore and lower the actual complexity.

Performance Limitations. Although Algorithm 2 introduces two optimization strategies, namely reducing the scale of the input graph and setting boundary conditions, the number of branch instances that still need to be explored in practice remains significant. Particularly when dealing with large-scale hypergraphs, the efficiency of the algorithm is still insufficient. This is primarily attributed to the following three limiting factors.

Lager Candidate Set: In Algorithm 2, the candidate set P comprises all the hyperedges in the input hypergraph. During each branching iteration, the size of P is decreased by only one (Lines 14-16), a process that is markedly inefficient. It has been noted that the hyperedges in set C influence those in P. Due to restrictions based on interaction frequency and interaction strength, certain hyperedges in P cannot be incorporated into C. In such instances, these hyperedges can be eliminated from P, effectively reducing the size of the candidate set.

Lack of Lower Bound Pruning: In Algorithm 2, a branch concludes only when the set P is empty or the branch already satisfies *HCMCS* constraints. However, this termination criterion is not particularly efficient. It has been observed that for a given set C, one can predict the size of the minimum sub-hypergraph that encompasses C and satisfies *HCMCS* constraints. Consequently, if the minimum required sub-hypergraph size for a set C surpasses the size of the current optimal solution, this branch can be preemptively terminated. This strategy significantly reduces the search space.

Trivial Branching Rules: In Algorithm 2, branch generation is based on randomly selecting a hyperedge from the set P, resulting in two diverging branches: one that includes and another that excludes the selected hyperedge. However, the criteria for branch selection substantially influence the speed of result generation. If the chosen hyperedge allows Algorithm 2 to swiftly achieve a smaller intermediate sub-hypergraph, then using the lower bound pruning mechanism, this approach can facilitate a quicker convergence to the optimal solution.

5 OPTIMIZATIONS FOR BRANCH-AND-BOUND

In this section, we introduce a branch-and-bound algorithm based on optimization strategies. The algorithm aims to address the problems identified in the previously proposed baseline algorithm: numerous branching instances are not effective, failing to generate outputs that surpass the existing results. In light of this, we have designed three optimization strategies: candidate set pruning, lower bound pruning, and branching strategy.

5.1 Candidate Set Pruning

In this subsection, we present a candidate set pruning technique tailored for branch instances (C, P). This technique is designed to substantially decrease the candidate set size by systematically eliminating elements from P that definitively cannot be incorporated into the set C. The implementation of this technique is primarily based on the definition of the (k, s)-HCore, which requires that each hyperedge intersects with at least k other hyperedges, and the strength of these interactions must be at least s. Following this, we first present the basic formalized equation for pruning, followed by a detailed analysis of the underlying principles.

Theorem 1. *(Basic Candidate Set Pruning) Given an instance (C, P), IE k and IS s. For any hyperedges $e \in P$, if $RIE_{C\cup P}(s, e) < k$, then we can discard e from P.*

Correctness Analysis. Clearly, the validity of Theorem 1 is self-evident based on the definition of the (k, s)-HCore. Furthermore, we define *minR* as the currently minimum community satisfying the *HCMCS* within our algorithm. This allows us to formulate a more refined pruning condition, one that is contingent on the size of the current minimum community.

Theorem 2. *(Candidate Set Pruning) Given an instance (C, P), IE k and IS s. For any hyperedges $e \in P$, if $\min\{RIE_{C\cup P}(s, e), RIE_{C\cup\{e\}}(s, e) + |minR| - |C| - 1\} < k$, then we can discard e from P.*

Correctness Analysis. Firstly, we prove $RIE_{C\cup P}(s, e) < k$ according to Theorem 1. Then, we analyze the process of incorporating hyperedge e into C. The inclusion of e needs to satisfy both the minimum community size (*minR*) and the interaction frequency criteria specific to e. In theory, the maximum number of hyperedges that can be added to C is $|minR| - |C| - 1$. When the candidate hyperedge e is added to the set C, it is inferred that the inclusion of e in C is not advisable, if we observe that $RIE_{C\cup\{e\}}(s, e) + |minR| - |C| - 1 < k$ This is because the inclusion of hyperedge e and the subsequent addition of hyperedges will result in a community size exceeding the *minR* limit. Consequently, this situation requires the exclusion of e in P. Therefore, based on the (k,s)-HCore constraint and the *minR*, it can be concluded that if $RIE_{C\cup\{e\}}(s,e) + |minR| - |C| - 1 < k$, then hyperedge e fails to satisfy the inclusion criteria for community C and should be pruned accordingly.

5.2 Lower Bound Pruning

In this subsection, we focus on introducing a method for calculating the lower bound. The key of this method is as follows: for a given instance (C,P), we can determine the size of the minimum community that includes the set C and satisfies the *HCMCS* constraint, denoted as $minSize(C)$. This allows us to effectively prune the instance P, especially when its lower bound exceeds the size of the *minR*. Algorithm 3 details this process, which iteratively refines and augments the community C.

Algorithm 3 Lower Bound

```
Input: Instance $(C,P)$, positive integers IE $k$, IS $s$
Output: Lower Bound $lb$
1: $r(e) \leftarrow k - RIE_C(s,e), \forall e \in C$;
2: $lb \leftarrow |C|$;
3: while $r(e) > 0, \exists e \in C$ do
4:     Select the hyperedge with the max $r(e)$;
5:     Select $r(e)$ hyperedges from $P$, denoted as $\Delta$, and the hyperedges
       intersect with the hyperedges in $C$ as much as possible;
6:     $lb \leftarrow lb + r(e)$
7:     $r(e) \leftarrow r(e) - |\{e' \in \Delta | e \cap e' \neq\}|, \forall e \in C$;
8: return $lb$;
```

Initially, the algorithm computes the residual interaction frequency requirement, denoted $r(e)$, for each hyperedge e within the community C. It then sets the lower bound, referred to as lb, to correspond to the size of C. In its main loop, the algorithm identifies the hyperedge e with the highest residual interaction frequency requirement. It selects $r(e)$ hyperedges from the candidate set P, which intersect as much as possible with the existing hyperedges in C. Subsequently, it updates the lower bound, lb. After each expansion, the residual interaction requirements of all hyperedges in C are updated. This iterative process continues until there are no longer any hyperedges in C with a residual interaction requirement exceeding zero. The final lower bound lb reflects the potential size of the minimum community that includes the set C and satisfies the *HCMCS* constraint.

Correctness Analysis. Subsequently added hyperedges are only guaranteed to satisfy the restricted interaction engagement constraint for the hyperedges within C, without considering whether the subsequently added hyperedges meet this constraint. Thus, the lower bound lb that we calculate must be the minimum size of any community that contains C and satisfies the *HCMCS* constraint.

Time complexity. The worst time complexity of Algorithm 3 is $O(|C|^2 + |C||P|)$. The time complexity analysis of Algorithm 3 begins with the initialization step. This step includes the computation of $RIE_C(s,e)$ for each hyperedge e in the community C, which has a complexity of $O(|C|)$. During the main loop, identifying the hyperedge with the highest residual interaction frequency, $r(e)$, necessitates traversing through C, thereby resulting in a complexity of $O(|C|)$. The process of selecting hyperedges from the candidate set P involves traversing the entire set, leading to a complexity of $O(|P|)$. Furthermore, updating C and recalculating $r(e)$ also requires a time complexity of $O(|C|)$. Consequently, the complexity for each iteration of the loop is $O(|C|+|P|)$. Given that the loop can execute up to $|C|$ times, the overall time complexity of the algorithm amounts to $O(|C|^2 + |C||P|)$.

Space complexity. The worst space complexity of Algorithm 3 is $O(|C| + |P|)$. We consider the storage requirements for the input sets C and P. Specifically, the space dedicated to set C aligns with the quantity of hyperedges, leading to a complexity of $O(|C|)$. Set P exhibits a similar trend, with its complexity noted as $O(|P|)$. During the execution of the algorithm, we maintain a set Δ to store the selected hyperedges from P that intersect with the hyperedges in C. In the worst case, Δ could be as large as P, hence its space requirement is also $O(|P|)$. Additionally, the algorithm maintains a counter $r(e)$ for each element e in the set C, and since $r(e)$ assigns an integer for each hyperedge, its space complexity is also $O(|C|)$. Summarizing these individual space complexities, the worst space complexity is $O(|C|) + O(|P|)$.

5.3 Branching Strategy

In this subsection, we present two branching strategies aimed at selecting hyperedges to be added to C. The primary goal of this strategy is to quickly identify a smaller community that meets the *HCMCS* constraint conditions, facilitating subsequent pruning of the candidate set and lower bound calculations. To achieve these goals, we give priority to those hyperedges that intersect with hyperedges of lower restricted interaction engagement within C, and also have a significant number of intersections with many hyperedges in C. Such a branching strategy helps Algorithm 2 to quickly identify a smaller community that meets the *HCMCS* constraint conditions. To quantitatively evaluate the preference for hyperedges, we introduce two connectivity score metrics for hyperedge e.

We propose the concept of the first connection score. In this connection score, we consider that when a hyperedge is added to the set C, it should intersect with the hyperedge with a lower *RIE* in the set C as much as possible. Hyperedge e that is associated with a greater number of hyperedges with lower *RIE* has a higher connectivity score. Therefore, it is given priority during the selection process for inclusion in the set C.

Definition 6. *(connection score) Given an instance (C,P), IE k and IS s. The connection score of a hyperedge $e \in P$ is defined as:*

$$\delta(e) = \sum_{e' \in NE_{C\cup\{e\}}(e)} \frac{1}{RIE_C(s,e')}. \tag{4}$$

We present another definition of the connectivity score below. We have considered two main aspects: $\delta_1(e)$ denotes the number of hyperedges e' in set C that are adjacent to e and have the $RIE_C(s,e')$ less than the threshold k; $\delta_2(e)$ represents the number of hyperedges e' that are adjacent to e and have the $IS(e,e')$ greater than or equal to the threshold s. The total connection score of hyperedge e, $\delta(e)$, is the sum of these two values. This definition quantifies the connectivity score of a hyperedge that is added to the set by integrating two dimensions.

Definition 7. *Given an instance (C,P), IE k and IS s. The connection score of a hyperedge $e \in P$ is defined as:*

$$\begin{aligned} \delta_1(e) &= |\{e' | e' \in NE_C(e), RIE_C(s,e') < k\}| \\ \delta_2(e) &= |\{e' | e' \in NE_C(e), IS(e,e') \geq s\}| \\ \delta(e) &= \delta_1(e) + \delta_2(e). \end{aligned} \tag{5}$$

The computation methods for these two connectivity scores give priority to hyperedges that intersect with those having the lower *RIE* and also intersect with many other hyperedges in the set C. This prioritization strategy effectively facilitates the rapid identification of smaller communities that meet the *HCMCS* constraint conditions, greatly enhancing the efficiency of pruning the candidate set and calculating lower bound pruning.

Correctness Analysis. The proposed branching strategy, including the connection score definitions, is designed to guide the search process toward smaller communities more efficiently. However, it is important to note that this selection strategy does not affect the correctness of the algorithm's results. The algorithm guarantees correctness because it exhaustively explores all feasible hyperedge combinations that satisfy the *HCMCS* constraints. The branching strategy merely prioritizes hyperedges based on their connectivity scores to expedite the identification of a valid community, but it does not exclude any feasible solutions from consideration. Thus, regardless of the order in which hyperedges are explored, the algorithm ultimately identifies the smallest community that meets the given constraints, ensuring the correctness of the result.

5.4 Optimization-based Branch-and-Bound

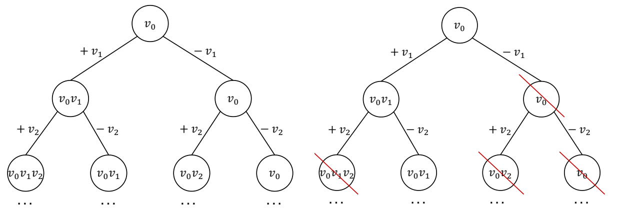
In this subsection, we integrate all the proposed optimization strategies to introduce an optimization-based branch-and-bound algorithm. The aim is to search for the minimum community that satisfies the HCMCS constraints.

Algorithm 4 OBBAB: Optimization-based BranchAndBound

```
Input: Hypergraph $G = (V, E)$, query vertex $q$, positive integers IE $k$, IS $s$
Output: Sub-hypergraph $S$
1: $cE(\cdot) \leftarrow (k, s)$-HCore Decomposition;
2: $G \leftarrow G \setminus \{e|cE(e) < k\}$;          ▷ Reduce the size of the hypergraph
3: if $G$ is empty then                               ▷ No community meeting the criteria
4:     return $\emptyset$;
5: $S \leftarrow G$;                                    ▷ Optimal result initialization
6: for $e \in NV_G(q)$ do
7:     $P \leftarrow E(G)$;                             ▷ Candidate set initialization
8:     BranchAndBound($\{e\}, P \setminus \{e\}$);
9: return $S$;
10: procedure BranchAndBound($C, P$)
11:     if $minSize(C) \geq |S|$ then                  ▷ Lower Bound Pruning
12:         return ;
13:     if $C$ is $(k, s)$-HCore and $|C| <= |S|$ then
14:         $S \leftarrow C$;
15:     else
16:         Prune $P$;                                  ▷ Candidate Set Pruning
17:         $e \leftarrow$ hyperedge with the highest score in P'; ▷ Branching Strategy
18:         BranchAndBound($C \cup \{e\}, P' \setminus \{e\}$);
19:         BranchAndBound($C, P' \setminus \{e\}$);
```

Algorithm 4 is an optimized branch-and-bound strategy aimed at finding the smallest community in a hypergraph that satisfies the HCMS constraints. The algorithm starts with a (k, s)-HCore decomposition to initialize the $cE(\cdot)$ (Line 1), followed by pruning to reduce the size of the hypergraph G by removing hyperedges with the $cE(\cdot)$ less than k (Line 2). If the pruned hypergraph is empty, indicating that no community meets the criteria, the algorithm returns an empty set (Lines 3-4). If the hypergraph is not empty, the algorithm sets the current hypergraph G as the initial optimal sub-hypergraph S (Line 5). For each adjacent hyperedge e of the query vertex q, the algorithm initializes the candidate set P (Lines 6-7) and performs the branch-and-bound process on it (Line 8). In the $BranchAndBound(C, P)$ function (Line 10), if the lower bound of the set C is not less than S (Line 11), candidate set pruning is performed and the current branch is terminated (Line 11-12). If C satisfies the (k, s)-HCore and its size does not exceed S (Line 13), then C is set as the new S. Otherwise, the algorithm prunes the current candidate set P according to Theorem 2 and selects the hyperedge e with the highest connectivity to proceed to the next round of branching (Lines 16-19). In summary, Algorithm 4 integrates candidate set pruning, lower bound pruning, and branching strategies to efficiently determine the minimum community.

Correctness Analysis. Since any technology used in Algorithm 4 has passed the correctness proof, it is obvious that Algorithm 4 is correct.



(a) The workflow of BAB (b) The workflow of OBBAB

Figure 2: The Workflow of BAB and OBBAB

EXAMPLE 2. *In Figure 2, we illustrate the workflows of the BAB and OBBAB algorithms. While the BAB algorithm exhaustively explores every branch, the OBBAB algorithm employs pruning techniques to eliminate unnecessary branches, reducing the number of traversed branches and improving efficiency.*

6 EXPERIMENT

In this section, we first describe the datasets and settings in our experiments. The experimental goals and results are then presented to demonstrate the effectiveness of the models and the efficiency of the algorithms.

Datasets. We use ten real-world datasets (NDCC, NDCS, TaMS, TaAU, ThAU, ThMS and DBLP[9], CoMH and CoGe [49], Aminer [31]) from six domains with various data properties. NDCC and NDCS [9] are two hypergraphs originating from the pharmaceutical field. NDCC comes from the National Drug Code, in which hyperedges represent drugs, and vertices are category labels used to tag drugs. NDCS comes from drug identifiers, with vertices representing substances, and hyperedges signifying drugs containing these substances. TaMS and TaAU [9] are hypergraphs with vertices being labels, and hyperedges being the sets of labels applied to questions on math.stackexchange.com and askubuntu.com. ThAU and ThMS [9] are hypergraphs that represent user activities on askubuntu.com and math.stackexchange.com. Each hyperedge captures threads engaged by users within 24 hours, and vertices correspond to individual users. CoMH and CoGe [49] are hypergraphs describing temporal evolution, simulating publishing activities of authors in the "History" and "Geology" fields in the Microsoft Academic Graph. In these two hypergraphs, vertices represent individual authors, and hyperedges correspond to their co-authorships within a specific time frame in given publications. DBLP [9] is a temporal higher-order network, where nodes are authors and each simplex represents a publication. AMiner [31] is a weighted undirected coauthorship

Table 1: Real-World Hypergraph Datasets.

DataSet	$\lvert E\rvert$	$\lvert V\rvert$	c_{max}	c_{avg}	d_{max}	d_{avg}
NDCC	46285	1149	24	3	5357	132
NDCS	29810	3767	25	7	5901	38
TaMS	558272	1627	5	2	59277	945
TaAU	219076	3021	5	3	19631	225
ThAU	117764	90054	14	2	2247	3
ThMS	563710	153806	21	2	12403	9
CoMH	308934	503868	925	2	1077	1
CoGe	1045462	1091979	25	3	1125	3
DBLP	1836596	2955129	20	4	1399	5
Aminer	27850748	17120546	18	4	9386	6

graph, where nodes are authors and edge weights correspond to the number of papers coauthored by two authors.

All datasets can be downloaded from ARB[1]. We remove all isolated vertices in the datasets. The basic statistics of the datasets are shown in Table 1, which is classified based on the different domains of origin for the hypergraphs. For each hypergraph, we define d_{max}, d_{avg} as the maximum and average degree of all vertices, and c_{max}, c_{avg} as the maximum and average cardinality of all hyperedges, respectively.

Settings. Our algorithms are implemented in C++ and compiled using the GNU GCC 11.3.0 compiler. The experiments are performed on a machine equipped with an AMD Ryzen Threadripper PRO 5995WX processor, featuring 64 cores running at 2.7 GHz, and 256 GB of memory. The operating system is Ubuntu 22.04.1 LTS.

Baseline Models. To demonstrate the precision of (k, s)-HCore in searching hypergraph communities, we compare it with several common hypergraph cohesive community models.

- k-hypercore [42]: each vertex is contained in at least k hyperedges.
- Nbr-k-core [5]: each vertex has at least k neighbors.
- *CoCore* [44]: each vertex is in at least k hyperedges and has h neighbors.

Experiment Evaluation. To demonstrate the advantages of our model and algorithm, we conduct the following experiments:

- **Model Evaluation:** We evaluate the (k, s)-HCore model by assess the hyperedge core number distribution to verify model robustness, compare overlapness with traditional models to demonstrate improved cohesiveness, analyze the maximum k value compared to vertex-centric models to highlight the ability to capture higher-order interactions, conduct case studies to showcases the practical applicability and flexibility of the (k, s)-HCore model in diverse real-world scenarios, and test the effectiveness of the query results produced by the model.
- **Algorithm Evaluation:** For the algorithm, we perform a query efficiency evaluation to assess the efficiency of the OB-BAB algorithm in comparison to the BAB method and evaluate its scalability with increasing dataset size.

6.1 Hyperedge Core Number Distribution

We conduct a comprehensive analysis of the hyperedge core numbers distribution of ten hypergraphs from six different domains. We employ a heatmap to elucidate how various interaction strengths impact the core decomposition outcomes. Figure 3 features individual subplots that delineate the core distribution for each dataset independently, with the x-axis signifying varying interaction strengths and the y-axis corresponding to hyperedge core numbers. The color gradient represents the percentage of hyperedges whose core numbers are less than or equal to the corresponding value on the y-axis. We set the interaction strength ranging from 2 to 5 for the TaMS and TaAU datasets, and from 2 to 10 for the remaining datasets, to align with the limitations imposed by hyperedge size.

Figure 3 demonstrates a noticeable negative correlation between the interaction strength and the maximum core number of the hypergraph. As the interaction strength increases, the maximum core number decreases. This empirical finding aligns with our expectations, as a higher interaction strength restricts the number of hyperedges that can satisfy the interaction requirements. A significant proportion of hyperedges fail to meet these strength parameters, resulting in a decrease in the maximum core number of hypergraphs. By adjusting the interaction strength, we can effectively control the structure of communities within the hypergraph. Figure 3 shows that in the NDCC and NDCS datasets, lower interaction strengths encourage the formation of cohesive communities with larger core numbers. Meanwhile, in the CoMH and CoDB datasets, higher interaction strengths facilitate the identification of more cohesive communities within large-scale hypergraphs.

6.2 Comparison of Overlapness

We evaluate the overlapness metric to gain insights into the density of interconnections within sub-hypergraphs of the hypergraph. The overlapness of a sub-hypergraph $H = (V_H, E_H)$ is calculated using the formula:

$$Overlapness(H) = \frac{\sum_{e \in E_H} |e|}{|V_H|}.$$

This metric measures the average number of hyperedges incident on each vertex within the sub-hypergraph, providing a quantitative assessment of how densely interconnected the hyperedges are. A higher overlapness value indicates a more densely interconnected subgraph, suggesting a tighter and more cohesive structure. Since k-core only has k, we do not analyze the changes in s for k-core. For *CoCore*, we use the s value as the h value to maintain consistency in the results.

Figure 4 presents the overlapness metrics of sub-hypergraphs using different cohesive modes ((k, s)-HCore, k-core, and *CoCore*). The results show that with the increase of k and s, the overlapness of (k, s)-HCore significantly increases, indicating that this method can identify more tightly and densely connected communities. In comparison, the overlapness of the k-core method is relatively low, and its increase is slower with the core number. *CoCore*'s performance falls between the other two methods. These results suggest that the (k, s)-HCore model has an advantage in identifying cohesive communities and functional groups within hypergraphs.

6.3 Comparison of Maximum k Value

In this section, we conduct a comparative analysis of different cohesive subgraph models, evaluating their performance on various datasets in terms of the maximum k value, as depicted in Figure 5. An increase in the maximum k value indicates a model's enhanced capability to detect a greater number of cohesive subgraphs. The s parameter of the (k, s)-HCore model significantly impacts the results: with s set to 2, the $(k, 2)$-HCore model consistently achieves high k values across all datasets, proving its

[1]https://www.cs.cornell.edu/~arb/data/

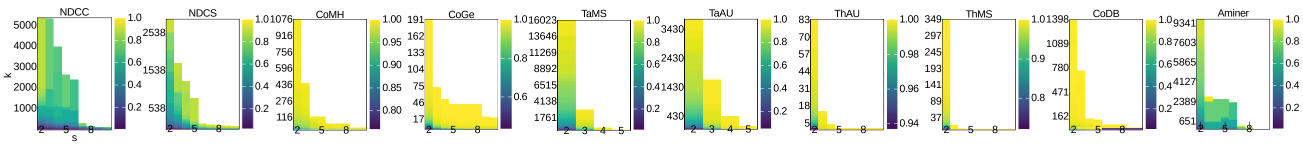


Figure 3: Cumulative percentage of hyperedge core numbers for 8 hypergraphs from 4 different fields.

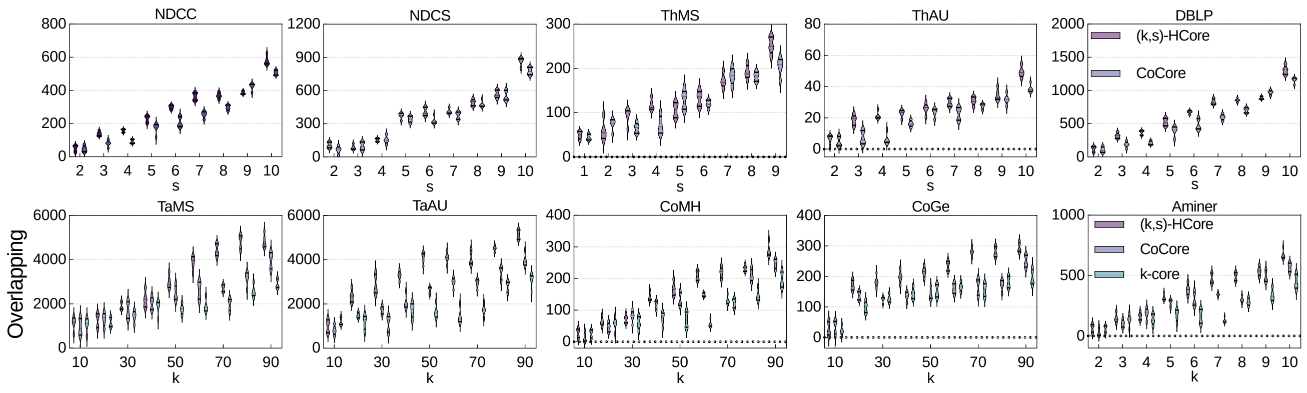


Figure 4: The (k, s)-HCore model over baseline vertex-centric cohesive subgraph models in terms of overlapness.

effectiveness in identifying cohesive subgraphs with numerous vertices. When s is increased to 5, there is a noticeable decrease in the maximum k value for the $(k, 5)$-HCore model. However, this trend reflects the imposition of more stringent constraints, each hyperedge must be at least 5 in length, and must intersect with other hyperedges by at least 5 vertices. Despite these constraints, the (k, s)-HCore model can identify more cohesive subgraphs, especially evident on the TaMS and TaAU datasets.

Thus, we ascertain that the (k, s)-HCore model, in comparison to traditional vertex-centric models, exhibits potential advantages in mining large-scale cohesive sub-hypergraphs with significant intersection density.

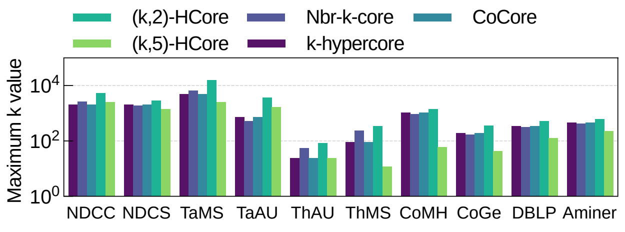


Figure 5: The (k, s)-HCore model over baseline vertex-centric cohesive subgraph models in terms of maximum k value.

6.4 Query Efficiency Evaluation

Figure 6 presents query efficiency across multiple datasets as the k and s parameters. The figure compares the BAB (traditional algorithm) with OBBAB (our algorithm), analyzing their performance in terms of query time under various k and s value settings. It is important to note that the query points are randomly selected for each dataset, ensuring a diverse and representative evaluation of both algorithms' query performance.

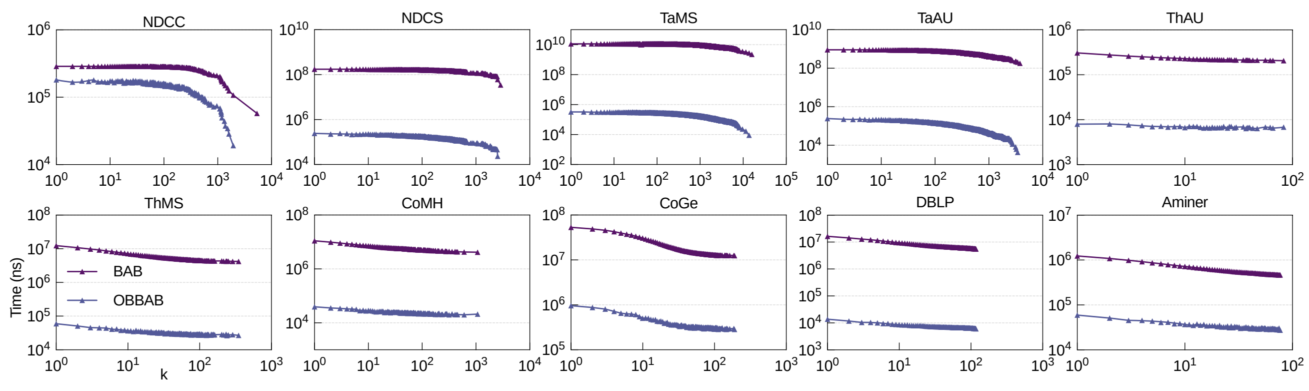
In Figure 6(a), with s set to 5, it is observed that as the hyperedge core number k increases, the query time for both algorithms tends to decrease. Meanwhile, the query times for the OBBAB algorithm are consistently lower across all datasets compared to the BAB algorithm, especially at larger values of k, indicating a more pronounced efficiency advantage of the optimized algorithm in processing large-scale cohesive subgraph queries.

Furthermore, in Figure 6(b), with k set to 50, an increase in the interaction strength s results in decreased query times. Contrary to the trend with k, the rise in s values decreases the search space due to the introduction of stricter interaction constraints, leading to improved query efficiency. The OBBAB algorithm demonstrates lower query times across all s values when compared to the BAB algorithm, further confirming the significant enhancement in query efficiency achieved by the optimized algorithm.

In summary, the experimental results distinctly demonstrate the significant optimization of the OBBAB algorithm in query time compared to the BAB algorithm, whether with increasing k values or s values. These findings underscore the critical role of algorithm optimization in enhancing the efficiency of detecting cohesive subgraphs within complex networks, particularly when dealing with large-scale network data.

6.5 Ablation Study

This subsection utilizes ablation analysis to evaluate the impact of three optimization strategies (lower bound pruning, candidate set pruning, and branching strategy) on algorithm performance. We compared the traditional algorithm (BAB), individual optimization strategies, and the fully optimized algorithm (OBBAB) in terms of query efficiency across multiple datasets, using the same vertex, k, and s parameters. Each experiment was repeated 100 times, and the average results were reported. The findings in Figure 7 indicate that the traditional algorithm exhibits consistently higher query times, while individual optimizations improve performance to varying degrees depending on the dataset characteristics. The OBBAB algorithm, which integrates all three optimizations, consistently achieves the best query efficiency across all datasets, demonstrating performance improvements


(a) Query time changes with k

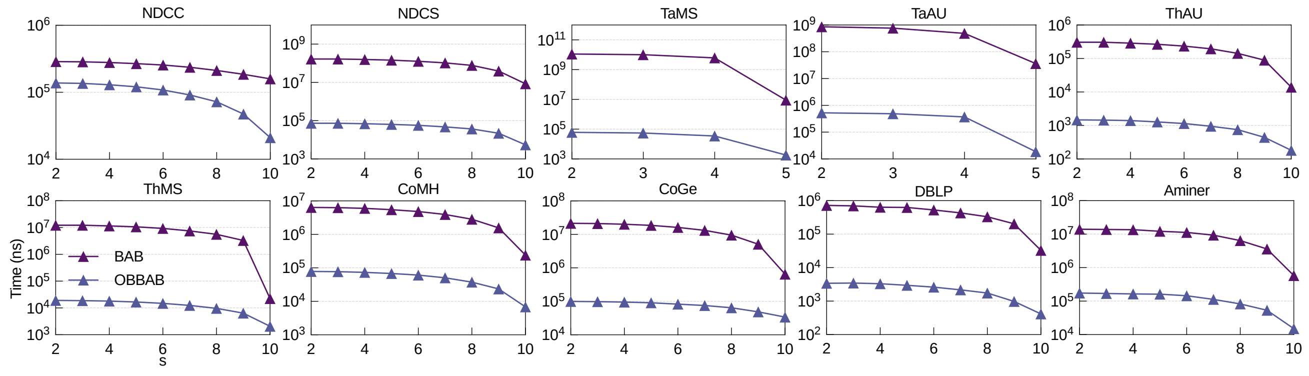

(b) Query time changes with s

Figure 6: Query efficiency of different algorithms.

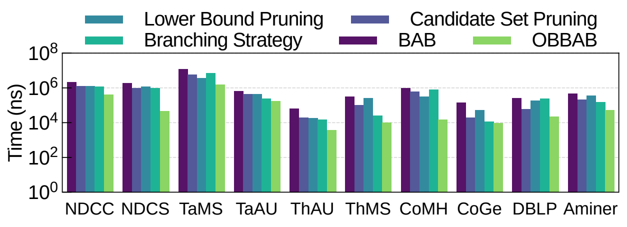


Figure 7: Ablation Study: query efficiency about optimization strategies

by orders of magnitude over the traditional algorithm. This highlights the effectiveness of the optimizations and the superiority of OBBAB.

6.6 Query Algorithm Scalability

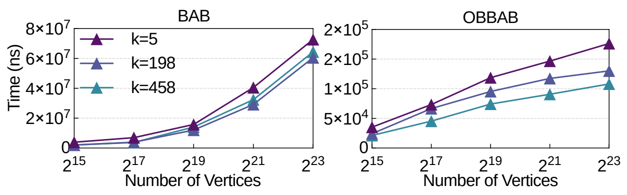


Figure 8: Relationship between algorithm query time and hypergraph size.

In this scalability experiment, we apply the existing synthetic hypergraph model [4] to create datasets with vertex sizes of 2^{15}, 2^{17}, 2^{19}, 2^{21}, and 2^{23}. For each subset, we conduct 100 repeated queries on a set of vertices with k values of 5, 198, and 458 and s value of 4, measuring the average time required for each query. The experiment compares the performance of the BAB algorithm with the OBBAB algorithm. Figure 8 shows that the time complexity of the BAB algorithm increases exponentially with the size of the dataset, whereas the OBBAB algorithm demonstrates an approximately linear relationship. Specifically, the BAB algorithm's query time increases exponentially as the dataset size increases, indicating poor scalability for larger datasets. In contrast, the OBBAB algorithm shows a near-linear increase in query time with the size of the dataset, demonstrating better scalability. These results suggest that the OBBAB algorithm is significantly more efficient and scalable for large hypergraph datasets.

6.7 Query Effectiveness Evaluation

To demonstrate the efficacy of our (k, s)-HCore model in CS, we compare the communities discovered by this model with those discovered by the baseline model. To this end, we select eight datasets, and set $s = 5$ in (k, s)-HCore configuration. Using the same vertices by selected randomly, we separately return the minimum communities with the maximum k values obtained by the two above models: To assess the quality of the returned communities, we utilize two community quality metrics.

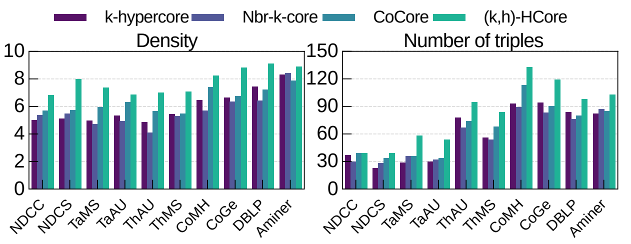


Figure 9: Evaluation of query effectiveness in the aspect of density and number of triples.

- **Community Density:** This parameter is a crucial metric for evaluating cohesion within a community. Typically, an excellent community should demonstrate a strong degree of internal linkage. An increase in density value signifies more intense connections within the community. This density is computed by the following formula:

$$Density = \frac{|\{e|e \in S\}|}{|\{v|v \in S\}|}.$$

- **Number of Triples (vertex set):** This parameter primarily measures the frequency of interaction within the community. An ideal community is characterized by rich internal interactions, which are reflected in a large number of vertex sets. The greater the number of triples (three vertices, vertex set) appearing in more than two hyperedges within a community, the higher the interaction frequency of that community. The number of triples is computed by the following formula:

$$NT = |\{(v_0, v_1, v_2)||\{e \in S|(v_0, v_1, v_2) \subseteq e\}| \geq 2\}|.$$

In the evaluation shown in Figure 9, the (k, h)-HCore model clearly exhibits superior performance in both density and number of triples metrics. Regarding density, the (k, h)-HCore model outperforms the other algorithms in most cases, particularly when compared with the k-hypercore and Nbr-k-core. Similarly, for the number of triples, (k, h)-HCore also shows an advantage, and even though the *CoCore* model demonstrates similar higher numbers of triples in NDCC, the overall trend still indicates that the (k, h)-HCore model is capable of maintaining high numbers of triples while also sustaining high density. This indicates that the (k, h)-model algorithm offers a more optimized query effectiveness in these two aspects, especially when dealing with complex network queries that require consideration of both structural density and the quantity of connections.

6.8 Case and Applications Studies

We apply three case studies and applications related to (k, s)-HCore. The **community search** demonstrates the differences between the communities identified by our algorithm and those identified by other algorithms. **Influence Spreading** and **hypergraph classification** highlight the advantages of our (k, s)-core over other types of cohesive subgraphs.

(1) **Community Search:** We employ the CoMH dataset, specifically dedicated to historical publications, and set an interaction strength parameter $s = 4$ with A. Chaniotis as the query vertex to conduct a minimal CS. The results reveal a cohesive community comprising A. Chaniotis, R.S. Stroud, R.A. Tybout, and T. Corsten, forming a $(106, 4)$-HCore community. In contrast, traditional vertex-centric models (k-core) of community cohesive can only identify a 65-core community, primarily due to

Table 2: Case study of community property using different cohesive modes.

	(k, s)-**HCore**	k-**core**	*CoCore*
E	182	276	254
V	28	39	31
Overlapness	35.2876	18.6325	23.156

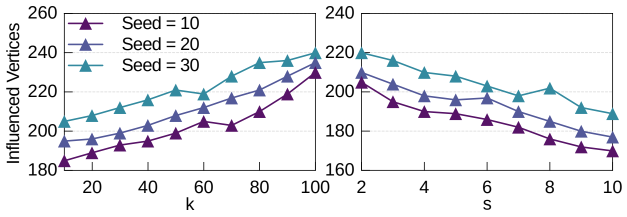


Figure 10: Evaluation of influence spreading for vertices in (k, s)-HCores.

interference from unrelated vertices. We measure the properties of the minimal sub-hypergraphs with A. Chaniotis as the query vertex through (k, s)-HCore ($s = 4$), k-core, and *CoCore* searches, as presented in Table 2. The results show that the $(k, 4)$-HCore method performed best in eliminating irrelevant vertices and identifying cohesive and relevant communities, whereas the k-core method is more susceptible to interference, resulting in lower cohesion and relevance. The *CoCore* method's performance fails between the two. A deeper analysis shows that these four scholars appeared together in 106 hyperedges, collaborating on a substantial volume of historical literature. According to Google searches, they significantly contribute to the Supplementum Epigraphicum Graecum (SEG), enriching its content and deepening the academic community's understanding of ancient Greek and Roman societies. This experiment demonstrates the (k, s)-HCore model's efficacy in focusing on core vertices, eliminating irrelevant vertex interference, and identifying more cohesive and relevant communities, highlighting its significance and applicability in academic research and other fields requiring precise community detection.

(2) **Influence Spreading:** To evaluate the vertex centrality of (k, s)-HCore, we utilized the SIR diffusion model described in [5, 30]. In our experiments, we randomly selected 10, 20, and 30 vertices as the initial infection sources. These vertices spread the infection to their adjacent vertices with a probability of 0.1 at each step. The diffusion process continued until no new vertices were infected. Existing methods for influence spreading often rely on vertex-centric measures, such as degree centrality or PageRank, which fail to capture higher-order group interactions within hypergraphs. These methods typically overlook the cohesive structures formed by hyperedges and the influence of their overlap, leading to suboptimal identification of influential vertices in hypergraph settings. In contrast, Figure 10 shows the results of selecting seed vertices in the (k, s)-HCores of NDCS to measure the number of infected vertices. We conducted 50 repetitions for each setting to ensure reliability. The results for the $(k, 5)$-HCore indicate that the average number of infected vertices increases as k increases. This phenomenon occurs because high k allows for the selection of more influential vertices. We also conducted experiments with the $(20, s)$-HCore, where the number of infected vertices increases with the number of seed vertices but decreases

as s increases within the (k, s)-HCores. This trend occurs because hyperedges smaller than s are excluded from the core, reducing volume density and vertex influence. Consequently, vertices in (k, s)-HCores with high k and low s exhibit greater influence.
(3) **Hypergraph Classification:** Hypergraphs from the same domain often exhibit similar characteristics in their overall hierarchical structure, such as the distribution of (k, s)-HCores and the range of s values. Traditional hypergraph classification methods usually focus on pairwise relationships or simple node features, failing to capture the rich, higher-order structures unique to hypergraphs. These approaches are often unable to effectively differentiate datasets with similar pairwise properties but distinct hyperedge-level interactions. Our experimental results validate the effectiveness of (k, s)-HCore distributions in hypergraph classification. In Figure 3, we show the core distribution of all datasets under the same (k, s)-core constraints. Notably, the core distributions of datasets from the same domain are highly similar. Additionally, the s value distributions of datasets from the same domain exhibit significant consistency. This observed similarity can serve as an effective preprocessing step in hypergraph classification tasks, providing insights into the unique structural patterns of hypergraphs that existing methods fail to exploit.

7 CONCLUSION

In this paper, we address the problem of community search in hypergraphs. Existing community models are predominantly vertex-centric, which suffer from the arbitrariness of hyperedge sizes and fail to capture adequately the intersection strength among hyperedges. To address these limitations, we propose a novel (k, s)-HCore model based on hyperedge interactions, effectively mitigating the issues of hyperedge size arbitrariness and providing a robust measure of intersection strength. Building on this model, we propose the *HCMCS* approach, aimed at identifying the minimal community within a hypergraph that meets the (k, s)-HCore constraint. Given the NP-hard nature of this problem, we develop a branch-and-bound algorithm, termed BAB, and further enhance its computational efficiency through three distinct optimization strategies, resulting in the optimized OBBAB algorithm. We conduct empirical evaluations across various different datasets and parameter configurations. The results demonstrate the advantages of our proposed algorithms in both efficiency and effectiveness. Additionally, we prove the superiority of (k, s)-HCore over traditional cohesive sub-hypergraph models from two key indicators: density and the number of triples. Furthermore, we demonstrate the benefits of our approach in diverse applications, such as community search, influence spreading, and hypergraph classification.

Future research will focus on developing more suitable cohesive sub-hypergraph models for hypergraph analysis and exploring more practical strategies and methods.

ACKNOWLEDGMENTS

This work was supported in part by Joint Key Funds of National Natural Science Foundation of China under Grant U23A20302, U24B20149.

REFERENCES

[1] Esra Akbas and Peixiang Zhao. 2017. Truss-based Community Search: a Truss-equivalence Based Indexing Approach. *Proc. VLDB Endow.* 10, 11 (2017), 1298–1309. https://doi.org/10.14778/3137628.3137640

[2] Esra Akbas and Peixiang Zhao. 2017. Truss-based Community Search: a Truss-equivalence Based Indexing Approach. *Proc. VLDB Endow.* 10, 11 (2017), 1298–1309. https://doi.org/10.14778/3137628.3137640

[3] Sinan G. Aksoy, Cliff A. Joslyn, Carlos Ortiz Marrero, Brenda Praggastis, and Emilie Purvine. 2020. Hypernetwork science via high-order hypergraph walks. *EPJ Data Sci.* 9, 1 (2020), 16. https://doi.org/10.1140/EPJDS/S13688-020-00231-0

[4] Naheed Anjum Arafat, Debabrota Basu, Laurent Decreusefond, and Stéphane Bressan. 2020. Construction and Random Generation of Hypergraphs with Prescribed Degree and Dimension Sequences. In *Database and Expert Systems Applications - 31st International Conference, DEXA 2020, Bratislava, Slovakia, September 14-17, 2020, Proceedings, Part II (Lecture Notes in Computer Science)*, Sven Hartmann, Josef Küng, Gabriele Kotsis, A Min Tjoa, and Ismail Khalil (Eds.), Vol. 12392. Springer, 130–145. https://doi.org/10.1007/978-3-030-59051-2_9

[5] Naheed Anjum Arafat, Arijit Khan, Arpit Kumar Rai, and Bishwamittra Ghosh. 2023. Neighborhood-based Hypergraph Core Decomposition. *Proc. VLDB Endow.* 16, 9 (2023), 2061–2074. https://doi.org/10.14778/3598581.3598582

[6] Nicola Barbieri, Francesco Bonchi, Edoardo Galimberti, and Francesco Gullo. 2015. Efficient and effective community search. *Data Min. Knowl. Discov.* 29, 5 (2015), 1406–1433. https://doi.org/10.1007/S10618-015-0422-1

[7] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O(m) Algorithm for Cores Decomposition of Networks. *CoRR* cs.DS/0310049 (2003). http://arxiv.org/abs/cs/0310049

[8] Ruben Becker, Federico Corò, Gianlorenzo D'Angelo, and Hugo Gilbert. 2020. Balancing Spreads of Influence in a Social Network. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020.* AAAI Press, 3–10. https://doi.org/10.1609/AAAI.V34I01.5327

[9] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon M. Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proc. Natl. Acad. Sci. USA* 115, 48 (2018), E11221–E11230. https://doi.org/10.1073/PNAS.1800683115

[10] Alain Bretto. 2013. Hypergraph theory: An Introduction. *An introduction. Mathematical Engineering. Cham: Springer* (2013).

[11] Lijun Chang, Jeffrey Xu Yu, Lu Qin, Xuemin Lin, Chengfei Liu, and Weifa Liang. 2013. Efficiently computing k-edge connected components via graph decomposition. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias (Eds.). ACM, 205–216. https://doi.org/10.1145/2463676.2465323

[12] I (Eli) Chien, Chung-Yi Lin, and I-Hsiang Wang. 2018. Community Detection in Hypergraphs: Optimal Statistical Limit and Efficient Algorithms. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain (Proceedings of Machine Learning Research)*, Amos J. Storkey and Fernando Pérez-Cruz (Eds.), Vol. 84. PMLR, 871–879. http://proceedings.mlr.press/v84/chien18a.html

[13] Gennaro Cordasco, Luisa Gargano, and Adele A. Rescigno. 2019. Active influence spreading in social networks. *Theor. Comput. Sci.* 764 (2019), 15–29. https://doi.org/10.1016/J.TCS.2018.02.024

[14] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. (2014), 991–1002. https://doi.org/10.1145/2588555.2612179

[15] Zheng Dong, Xin Huang, Guorui Yuan, Hengshu Zhu, and Hui Xiong. 2021. Butterfly-Core Community Search over Labeled Graphs. *Proc. VLDB Endow.* 14, 11 (2021), 2006–2018. https://doi.org/10.14778/3476249.3476258

[16] Anton Eriksson, Timoteo Carletti, Renaud Lambiotte, Alexis Rojas, and Martin Rosvall. 2022. Flow-based community detection in hypergraphs. In *Higher-Order Systems.* Springer, 141–161.

[17] Ernesto Estrada and Juan A Rodriguez-Velazquez. 2005. Complex networks as hypergraphs. *arXiv preprint physics/0505137* (2005).

[18] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *VLDB J.* 29, 1 (2020), 353–392. https://doi.org/10.1007/S00778-019-00556-X

[19] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *VLDB J.* 29, 1 (2020), 353–392. https://doi.org/10.1007/S00778-019-00556-X

[20] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2019. Effective and Efficient Community Search Over Large Directed Graphs. *IEEE Trans. Knowl. Data Eng.* 31, 11 (2019), 2093–2107. https://doi.org/10.1109/TKDE.2018.2872982

[21] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 6 (2020), 854–867. https://doi.org/10.14778/3380750.3380756

[22] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 6 (2020), 854–867. https://doi.org/10.14778/3380750.3380756

[23] Kasimir Gabert, Ali Pinar, and Ümit V. Çatalyürek. 2021. A Unifying Framework to Identify Dense Subgraphs on Streams: Graph Nuclei to Hypergraph Cores. (2021), 689–697. https://doi.org/10.1145/3437963.3441790

[24] Christos Giatsidis, Dimitrios M. Thilikos, and Michalis Vazirgiannis. 2011. D-cores: Measuring Collaboration of Directed Graphs Based on Degeneracy. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, Diane J. Cook, Jian Pei, Wei Wang, Osmar R.

Zaïane, and Xindong Wu (Eds.). IEEE Computer Society, 201–210. https://doi.org/10.1109/ICDM.2011.46

[25] Priya Govindan, Chenghong Wang, Chumeng Xu, Hongyu Duan, and Sucheta Soundarajan. 2017. The k-peak Decomposition: Mapping the Global Structure of Graphs. (2017), 1441–1450. https://doi.org/10.1145/3038912.3052635

[26] Yuanshen Guan, Xiangguo Sun, and Yongjiao Sun. 2023. Sparse relation prediction based on hypergraph neural networks in online social networks. *World Wide Web (WWW)* 26, 1 (2023), 7–31. https://doi.org/10.1007/S11280-021-00936-W

[27] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. (2014), 1311–1322. https://doi.org/10.1145/2588555.2610495

[28] Xin Huang, Laks V. S. Lakshmanan, and Jianliang Xu. 2017. Community Search over Big Graphs: Models, Algorithms, and Opportunities. (2017), 1451–1454. https://doi.org/10.1109/ICDE.2017.211

[29] Xin Huang, Wei Lu, and Laks V. S. Lakshmanan. 2016. Truss Decomposition of Probabilistic Graphs: Semantics and Algorithms. (2016), 77–90. https://doi.org/10.1145/2882903.2882913

[30] Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. 2010. Identification of influential spreaders in complex networks. *Nature physics* 6, 11 (2010), 888–893.

[31] Raunak Kumar, Paul Liu, Moses Charikar, and Austin R. Benson. 2020. Retrieving Top Weighted Triangles in Graphs. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang (Eds.). ACM, 295–303. https://doi.org/10.1145/3336191.3371823

[32] Dong Li, Zhiming Xu, Sheng Li, and Xin Sun. 2013. Link prediction in social networks based on hypergraph. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, Leslie Carr, Alberto H. F. Laender, Bernadette Farias Lóscio, Irwin King, Marcus Fontoura, Denny Vrandecic, Lora Aroyo, José Palazzo M. de Oliveira, Fernanda Lima, and Erik Wilde (Eds.). International World Wide Web Conferences Steering Committee / ACM, 41–42. https://doi.org/10.1145/2487788.2487802

[33] Rong-Hua Li, Jiao Su, Lu Qin, Jeffrey Xu Yu, and Qiangqiang Dai. 2018. Persistent Community Search in Temporal Networks. (2018), 797–808. https://doi.org/10.1109/ICDE.2018.00077

[34] Xuankun Liao, Qing Liu, Jiaxin Jiang, Xin Huang, Jianliang Xu, and Byron Choi. 2022. Distributed D-core Decomposition over Large Directed Graphs. *Proc. VLDB Endow.* 15, 8 (2022), 1546–1558. https://doi.org/10.14778/3529337.3529340

[35] Yu-Ru Lin, Jimeng Sun, Paul C. Castro, Ravi B. Konuru, Hari Sundaram, and Aisling Kelliher. 2009. MetaFac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki (Eds.). ACM, 527–536. https://doi.org/10.1145/1557019.1557080

[36] Boge Liu, Fan Zhang, Wenjie Zhang, Xuemin Lin, and Ying Zhang. 2021. Efficient Community Search with Size Constraint. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 97–108. https://doi.org/10.1109/ICDE51399.2021.00016

[37] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. VAC: Vertex-Centric Attributed Community Search. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 937–948. https://doi.org/10.1109/ICDE48307.2020.00086

[38] Yu Liu, Qi Luo, Mengbai Xiao, Dongxiao Yu, Huashan Chen, and Xiuzhen Cheng. 2024. Reordering and Compression for Hypergraph Processing. *IEEE Trans. Computers* 73, 6 (2024), 1486–1499. https://doi.org/10.1109/TC.2024.3377915

[39] Linyuan Lu and Xing Peng. 2013. High-Order Random Walks and Generalized Laplacians on Hypergraphs. *Internet Math.* 9, 1 (2013), 3–32. https://doi.org/10.1080/15427951.2012.678151

[40] Linhao Luo, Yixiang Fang, Xin Cao, Xiaofeng Zhang, and Wenjie Zhang. 2021. Detecting Communities from Heterogeneous Graphs: A Context Path-based Graph Neural Network Model. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 1170–1180. https://doi.org/10.1145/3459637.3482250

[41] Qi Luo, Zhenzhen Xie, Yu Liu, Dongxiao Yu, Xiuzhen Cheng, Xuemin Lin, and Xiaohua Jia. 2024. Sampling hypergraphs via joint unbiased random walk. *World Wide Web (WWW)* 27, 2 (2024), 15. https://doi.org/10.1007/S11280-024-01253-8

[42] Qi Luo, Dongxiao Yu, Zhipeng Cai, Xuemin Lin, and Xiuzhen Cheng. 2021. Hypercore Maintenance in Dynamic Hypergraphs. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 2051–2056. https://doi.org/10.1109/ICDE51399.2021.00199

[43] Qi Luo, Dongxiao Yu, Zhipeng Cai, Xuemin Lin, Guanghui Wang, and Xiuzhen Cheng. 2023. Toward maintenance of hypercores in large-scale dynamic hypergraphs. *VLDB J.* 32, 3 (2023), 647–664. https://doi.org/10.1007/S00778-022-00763-Z

[44] Qi Luo, Dongxiao Yu, Yu Liu, Yanwei Zheng, Xiuzhen Cheng, and Xuemin Lin. 2023. Finer-Grained Engagement in Hypergraphs. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 423–435. https://doi.org/10.1109/ICDE55515.2023.00039

[45] Fragkiskos D. Malliaros and Michalis Vazirgiannis. 2013. To stay or not to stay: modeling engagement dynamics in social graphs. (2013), 469–478. https://doi.org/10.1145/2505515.2505561

[46] Leng Min. 2013. An O(m) Algorithm for Cores Decomposition of Undirected Hypergraph. *Journal of Chinese Computer Systems* (2013).

[47] Nicolò Ruggeri, Martina Contisciani, Federico Battiston, and Caterina De Bacco. 2023. Community detection in large hypergraphs. *Science Advances* 9, 28 (2023), eadg9159.

[48] Stephen B. Seidman. 1983. Network structure and minimum degree. *Social Networks* 5 (1983), 269–287.

[49] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (Eds.). ACM, 243–246. https://doi.org/10.1145/2740908.2742839

[50] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. (2010), 939–948. https://doi.org/10.1145/1835804.1835923

[51] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Jiuxin Cao, Yingxia Shao, and Nguyen Quoc Viet Hung. 2021. Heterogeneous Hypergraph Embedding for Graph Classification. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, Liane Lewin-Eytan, David Carmel, Elad Yom-Tov, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 725–733. https://doi.org/10.1145/3437963.3441835

[52] Yu Wang and Qilong Zhao. 2022. Multi-Order Hypergraph Convolutional Neural Network for Dynamic Social Recommendation System. *IEEE Access* 10 (2022), 87639–87649. https://doi.org/10.1109/ACCESS.2022.3199364

[53] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudré-Mauroux. 2019. Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach. (2019), 2147–2157. https://doi.org/10.1145/3308558.3313635

[54] Yixing Yang, Yixiang Fang, Xuemin Lin, and Wenjie Zhang. 2020. Effective and Efficient Truss Computation over Large Heterogeneous Information Networks. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 901–912. https://doi.org/10.1109/ICDE48307.2020.00083

[55] Xiaoyao Zheng, Yonglong Luo, Liping Sun, Xintao Ding, and Ji Zhang. 2018. A novel social network hybrid recommender system based on hypergraph topologic structure. *World Wide Web* 21, 4 (2018), 985–1013. https://doi.org/10.1007/S11280-017-0494-5

[56] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding. (2006), 1601–1608. https://proceedings.neurips.cc/paper/2006/hash/dff8e9c2ac33381546d96deea9922999-Abstract.html

[57] Zhongxin Zhou, Fan Zhang, Xuemin Lin, Wenjie Zhang, and Chen Chen. 2019. K-Core Maximization: An Edge Addition Approach. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 4867–4873. https://doi.org/10.24963/IJCAI.2019/676

[58] Gaoping Zhu, Xuemin Lin, Ke Zhu, Wenjie Zhang, and Jeffrey Xu Yu. 2012. TreeSpan: efficiently computing similarity all-matching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman (Eds.). ACM, 529–540. https://doi.org/10.1145/2213836.2213896