Deep Overlapping Community Search via Subspace Embedding

Qing Sima[1], Jianke Yu[2], Xiaoyang Wang[1], Wenjie Zhang[1], Ying Zhang[2], Xuemin Lin[3]

[1]University of New South Wales, Sydney, Australia

[2]University of Technology Sydney, Sydney, Australia [3]Shanghai Jiao Tong University, Shanghai, China

{q.sima,xiaoyang.wang1,wenjie.zhang}@unsw.edu.au,{jianke.yu@student,ying.zhang}@uts.edu.au,xuemin.lin@sjtu.edu.cn

ABSTRACT

Overlapping Community Search (OCS) identifies nodes that interact with multiple communities based on a specified query. Existing community search approaches fall into two categories: algorithm-based models and ML-based models. Despite the long-standing focus on this topic within the database domain, current solutions face two major limitations: 1) Both approaches fail to address personalized user requirements in OCS, consistently returning the same set of nodes for a given query regardless of user differences. 2) Existing ML-based CS models suffer from severe training efficiency issues. In this paper, we formally redefine the problem of OCS. By analyzing the gaps in both types of approaches, we then propose a general solution for OCS named **S**parse **S**ubspace **F**ilter (SSF), which can extend any ML-based CS model to enable personalized search in overlapping structures. To overcome the efficiency issue in the current models, we introduce **S**implified **M**ulti-hop Attention **N**etworks (SMN), a lightweight yet effective community search model with larger receptive fields. To the best of our knowledge, this is the first ML-based study of overlapping community search. Extensive experiments validate the superior performance of SMN within the SSF pipeline, achieving a 13.73% improvement in F1-Score and up to 3 orders of magnitude acceleration in model efficiency compared to state-of-the-art approaches.

1 INTRODUCTION

Identifying a closely interrelated community based on a query node is a long-standing focus within the database domain, facilitating various applications, including fraud detection [10, 30] and recommender systems [14, 18]. Existing Community Search (CS) models can be categorized into algorithm-based and ML-based approaches. Algorithm-based models define a community as a cohesive group of nodes [11, 15, 37], while ML-based approaches are task-driven which define communities using labels, or node types [6, 14, 18, 44]. Leveraging predictive capabilities, ML-based models identify a set of nodes closely related to the query sharing the same label [14]. However, these models cannot be extended to Overlapping Community Search (OCS). Therefore, this paper aims to develop efficient ML-based community search models for overlapping community structures, addressing the limitations of existing methods.

Overlapping community structure allows each node to interact with multiple communities, each exhibiting distinct characteristics such as sizes, levels of cohesiveness, and attribute patterns [4, 13, 53]. This complexity raises a key challenge in OCS: how do we prioritize or rank communities when a query node belongs to multiple at once? Given the diversity in user interests, it becomes crucial to offer users the flexibility to personalize their search by selecting which target communities they wish to focus on. Existing methods often fall short by returning the same community for a given query, ignoring the differences in user preferences. Therefore, enabling the interactive selection of target communities is essential to ensuring the results align with user requirements.

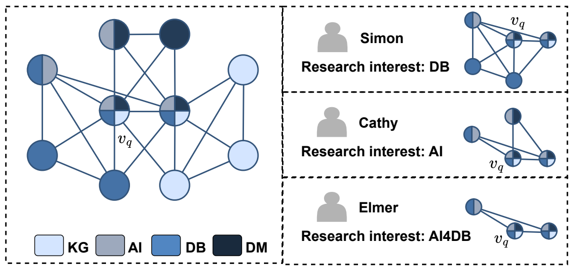
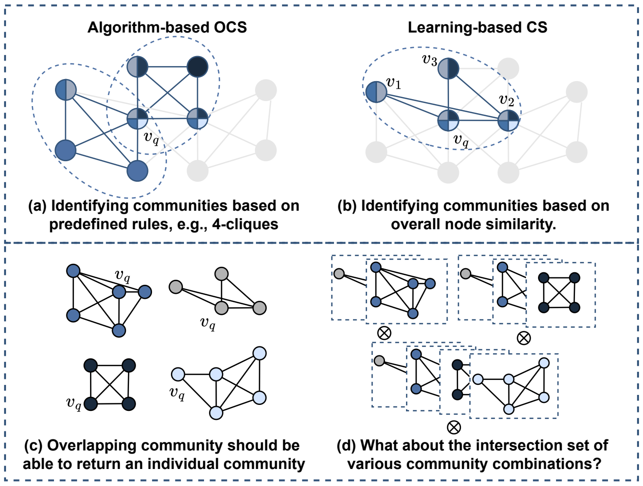


Figure 1: Different users are expecting different communities given the same query node

Figure 1 shows a toy example of citation networks, where nodes represent papers and edges denote citation relationships. The colors on the nodes indicate community affiliations, with multi-colored nodes representing papers overlapping multiple domains. Community search can serve as a tool that recommends related papers based on a user's current reading, i.e., the query node. The right part of the figure demonstrates how different users with distinct research interests seek personalized community recommendations based on the same query node v_q. Simon, Cathy, and Elmer have different research interests: DB, AI, and AI4DB, respectively. Each subgraph highlights a set of nodes belonging to a domain, represented by the underlying color. For example, users interested in the DB domain, like Simon, will specifically search for blue nodes. Particularly, Elmer's interest in AI4DB represents an intersection between multiple communities, requiring the model to identify nodes that fall into both target communities: AI and DB. Hence, it is essential to allow users to select target communities addressing their personalized needs. Compared to disjoint community search [14, 24, 30], Overlapping Community Search (OCS) is a more challenging problem and has not received much attention in existing ML-based literature. The study of OCS yields potential benefits across various applications. For example, it enables the precise extraction of fraudulent entities from multiple communities [52], the discovery of literature in the cross-domain [40], and the recommendation of products to the most valuable community [32].

Existing solutions. Popular algorithm-based approaches employ different structural constraints to measure subgraph cohesiveness, such as k-core [11, 15, 37], k-truss [1, 2, 21], and k-clique [4, 46, 53]. Algorithm-based overlapping community search models are designed to discover multiple subgraphs containing the query node, each meeting a certain level of cohesiveness requirements [4, 13, 53]. However, these models struggle to assign predictive labels like "DB" to each node, assuming all returned communities carry the same



Algorithm-based OCS
Learning-based CS
v_q
v_1
v_2
v_3
(a) Identifying communities based on predefined rules, e.g., 4-cliques
(b) Identifying communities based on overall node similarity.
(c) Overlapping community should be able to return an individual community
(d) What about the intersection set of various community combinations?

Figure 2: Challenges in existing approaches and user expectation given datasets with overlapping communities

semantics [14]. In addition, these approaches only capture linear attribute-wise patterns and tend to measure structural cohesiveness and attribute homogeneity independently [24].

Comparatively, ML-based community search models are task-orientated and identify communities by prior knowledge learned from limited labels [14, 24, 30, 45]. Current models are mainly built with Graph Neural Networks (GNNs) to learn node representations for online searching [16, 26, 41, 50], such as ICS-GNN [14], QDGNN [24] and COCLEP [30]. These models have two main phases, including offline training and online searching. The offline training stage focuses on learning representative node embeddings from a subset of labeled data. Online searching algorithms utilize pairwise GNN scores (probabilities) [14, 24] or node similarity [30] against the query to identify communities. However, current ML-based approaches primarily focus on disjoint community search, ignoring the nature that nodes tend to demonstrate various community affiliations. Therefore, two main motivations exist for designing an efficient and effective ML-based approach for OCS.

Motivation 1. *How to search for customized communities under overlapping community structures?* Although extensive work has been conducted, the aforementioned methods failed to address specified user requirements. Given the same query, both types of models consistently return the same set of nodes for different users [14, 25]. Figure 2(a) demonstrates the community identification process of an algorithm-based approach. Given the query v_q and a cohesiveness constraint, such as 4-clique [4], algorithm-based OCS models will return the highlighted communities by identifying two distinct subgraphs defined by 4-clique. This approach assumes that all cliques in the returned set share the same semantics, leading to two issues. First, the models fail to distinguish nodes from different cliques, requiring manual intervention. Additionally, cohesive subgraphs are unaware of downstream labels, and since not all nodes in the clique share the same label, the models struggle to exclude irrelevant nodes.

ML-based CS models measure the overall node similarity guided by node labels [14, 24, 30]. As illustrated in Figure 2(b), given the query v_q, it tends to return nodes that exhibit higher similarity levels (v_1, v_2, v_3). It is observable that the returned nodes will have closer embeddings by sharing at least two common communities with the query (each returned node shares at least two colors). To control the size of the returned set, these models utilize a threshold to gate the level of similarity [24, 30]. In this example, a relaxed threshold might include the entire graph, as all nodes share at least one label with the query shown in Figure 1. Thus, both algorithm-based OCS and ML-based CS failed to identify user-specified communities.

Motivation 2. *How to design an efficient and effective model framework for ML-based OCS?* Current ML-based models suffer from severe training efficiency issues. COCLEP [30] uses graph partition techniques to improve the model training efficiency. However, this approach has drawbacks such as loss of global context, boundary issues, and data imbalance. Moreover, the complexity of overlapping community structures necessitates a model that favors high-order awareness, requiring it to gather messages from large receptive fields. Therefore, designing an efficient model framework with larger receptive fields remains a significant challenge.

Define ML-based OCS. By identifying challenges when applying the existing model to overlapping communities, we redefine the OCS problem within the deep learning context. A well-trained OCS model should account for each user's specific interests, guiding the community search to identify personalized communities. Therefore, an ML-based OCS model needs first to predict the potential labels for the query node. It then allows users to interactively select the relevant labels as the target, thereby restricting the search to nodes within the specified target community to fulfill user requirements.

In contrast to the expected community identified by existing approaches illustrated in Figure 2(a) and 2(b), a qualified OCS model should be able to effectively retrieve those four pure communities individually, as depicted in Figure 2(c). For example, the model should only return the blue node-set representing papers related to the DB domain, i.e., the target community. This further raises a more challenging question, named Overlapping Communities Intersection Search (OCIS), illustrated in Figure 2(d). What if a user is interested in multiple domains?

For example, in citation networks (e.g., Figure 1), a researcher might seek papers that lie at the intersection of multiple prominent fields, such as AI and DB. Given the vast number of papers in both fields, retrieving all papers from each domain would be overwhelming. The target is to narrow the search to find papers that cover both fields, meeting the researcher's need for cross-domain insights and resulting in a more focused and relevant set of results. Similarly, in social networks, community intersection search helps identify users with overlapping interests or affiliations. For instance, if a user engages in both technology and entrepreneurship communities, they can serve as a query to find others with similar profiles. When explicit labels are not available, the model can use predicted memberships from shared connections and attributes to identify relevant users, enabling more personalized recommendations and deeper insights into the network.

To solve OCIS, a possible brute-force approach is first to identify all the target communities separately and then calculate their intersections. However, this transforms the task into an exhaustive enumeration of communities, which is impractical due to the high computational cost. This paper proposes a general solution to tackle the OCS problem. The technique developed is also effective for the OCIS scenario without the requirement of enumerating all communities, reducing the computational overhead.

Our solution. In this paper, we introduce a subspace community embedding technique called Sparse Subspace Filter (SSF) to tackle the challenge of identifying and segregating nodes with overlapping community affiliations. SSF is a general technique that can extend any existing ML-based model primarily built for the disjoint community search problem to OCS. Moreover, to address the limitations in previous approaches, we replace the existing model with a novel framework named Simplified Multi-hop Attention Network (SMN), which significantly improves the model training speed while preserving high-order awareness.

Sparse Subspace Filter. SSF aims to learn a sparse matrix, representing each community by a sparse embedding. This technique enables node embeddings to fall into multiple subspaces simultaneously, effectively identifying the target set under overlapping community structures. When searching for a target community, the learned sparse community embeddings are used as a basis vector to project nodes into the underlying subspace. The community search is then conducted exclusively within the target subspace. The proposed technique also extends any ML-based CS model to the scenario with multiple target communities, using the union subspace to represent the intersection of communities. SSF efficiently identifies the intersection between communities without the need to enumerate the entire graph for community affiliations.

Lightweight model framework, SMN. To address the challenges in existing ML-based CS models, we propose a novel model named SMN. By proving the training inefficiency in the popular graph-query frameworks, SMN adopts a simplified model structure to alleviate the burden. Moreover, as an OCS model needs a larger receptive field to capture high-order patterns, SMN uses an advanced hop-wise attention mechanism to cover higher-hop neighborhoods while preventing the model from oversmoothing.

Contributions. The main contributions of this paper are summarized as follows:

- To the best of our knowledge, we are the first to investigate the problem of overlapping community search in the ML-based scenario.
- A general solution, named SSF, is then proposed, which is effective in finding a pure community as well as handling the intersection scenario.
- Moreover, we introduce a Simplified Multi-hop Attention Network (SMN), which is efficient in model training while capturing high-order patterns.
- Extensive experiments on 9 overlapping and 4 disjoint community datasets show that our model can achieve an average F1-Score improvement over state-of-the-art methods of 13.73% and 7.62%, respectively. Additionally, our approach enhances model training efficiency by 3 orders of magnitude and online query efficiency by 2 orders of magnitude.

2 RELATED WORK

Algorithm-based community search. The community search problem is widely studied in the literature and can find many applications. Different cohesiveness metrics are leveraged, such as k-core [15, 37, 39], k-truss [1, 2, 21], and k-clique [4, 46, 53], which efficiently identify communities based on the graph structure. Moreover, researchers conduct studies on attributed graphs and extend their analysis by incorporating attribute constraints alongside structural considerations to identify a set of nodes with similar attributes [7–9, 35]. Additionally, several studies have focused on discovering communities that contain multiple query nodes [2, 20, 21, 37]. Given a set of query nodes, the studies aim to find a densely connected subgraph that contains all the query nodes. For example, CTC [21] and FirmTruss [2] are designed to search for the community including all the query nodes while satisfying different constraints, e.g., closest truss and firm truss. These studies are orthogonal to our research, focusing on overlapping communities and personalized community search. Algorithm-based OCS [4, 13, 53] enable the query node to possess multiple community affiliations with equivalent levels of cohesiveness, such as being part of two subgraphs that fulfill k-clique constraints. However, these models often return all communities containing the query nodes without the ability to focus on a specific community. Nonetheless, the lack of label awareness hampers these models' capacity to identify and separate nodes from distinct communities.

GNN-based community search. GNN and its variants have achieved considerable success in graph analytic tasks, including node classification [16, 26] and subgraph mining [42, 43]. The GNN model learns from predefined ground truth, effectively capturing patterns from node attributes while considering diverse graph structures simultaneously. In addition, advanced models have been introduced to improve model expressiveness and efficiency [41, 49, 50]. Recently, GNN-based community search models have attracted increasing attention due to their flexible structure constraints and expressive power. These models can distinguish nodes from different communities by balancing the contribution from both the topological structure and the nodes' attributes. Deep CS models are trained using prior knowledge, making their assumptions more realistic than traditional approaches. A community is identified by a group of nodes sharing similar patterns in attributes and topological structures. ICS-GNN [14] introduces an online deep community search model using a vanilla GCN model. The model is transductive, conducting training and online querying within the identified candidate subgraph. QDGNN [24] employs an offline setting by training the model on a fixed training set and inferring the model onto the unseen test set. The model extends to an attributed community search by adopting an attribute encoder to identify a group of nodes that contain a set of attributes. ALICE [45] focuses on attributed community search by combining a candidate subgraph extraction phase using density sketch modularity. The model follows the query-graph encoder frameworks and adopts a cross-attention encoder to control the interaction. COCLEP [30] follows the framework of QDGNN and conducts semi-supervised training by leveraging contrastive learning techniques. The model uses a hypergraph as an augmented graph and propagates information using GCN and Hyper GNN [12]. However, current models have struggled to adapt to the overlapping community search, often encountering issues with oversmoothing and slow training.

3 PRELIMINARIES

3.1 Problem Definition

Let $G = (V, E)$ be an undirected graph with a set V of nodes and a set E of edges. Let $n = |V|$ and $m = |E|$ be the number of nodes and edges, respectively. Given a node $u \in V$, $N(u) = \{v|(u, v) \in E\}$ is the neighbor set of u. The adjacency matrix of G is denoted as $\boldsymbol{A} \in \{0, 1\}^{n\times n}$, where $\boldsymbol{A}_{i,j} = 1$, if $(v_i, v_j) \in E$, otherwise $\boldsymbol{A}_{i,j} = 0$. $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ is the set of node features and $\boldsymbol{x}_i$ represents the node features of v_i. Given a query node q, the CS problem aims to find a k-sized set of nodes containing the query from G while maximizing the GNN score or node similarity against the query [14, 24, 30]. Under overlapping community structures, each node u belongs to more than one community, i.e., $u \in C_u = \{C_u^{z_1}, C_u^{z_2}, ..., C_u^{z_i}\}$, where C_u is the set of communities contains q, and $z_i \in Z$ is the label of a community. Users are allowed to select a target community label $t \in Z$ to guide the community search. Following the existing definition of community search in ML-based models [14, 24, 30], we define OCS as below:

Definition 3.1 (Overlapping Community Search, OCS). *Given a graph G, a query node q, a community size k, and a target community label $t \in Z$, OCS aims to identify a k-sized query-dependent group of nodes V_c that are closely intra-related. This group satisfy $V_c \subseteq C_q^t$ and $|V_c| = k$, where C_q^t is the true community associated with the target label t.*

Under this definition, the user is only interested in a single community, i.e., the target community. However, as discussed, due to the complexity of overlapping structures, describing the desired group using a single target community might not be adequate. Defining a more refined community by considering the intersection of multiple target groups is often preferable. Therefore, we extend OCS by introducing the following definition to enhance flexibility.

Definition 3.2 (Overlapping Communities Intersection Search, OCIS). *Given a graph G, a query node q, a community size k, and multiple target community labels $\boldsymbol{T_q} = \{t_1, t_2, ..., t_i\} \in Z$, OCIS aims to search for the user-specified intersection set V_c' of size k such that $V_c' \subseteq \hat{C}_q$, where $\hat{C}_q = C_q^{t_1} \cap C_q^{t_2} \cap ... \cap C_q^{t_i}$*

Within this definition, the intersection of multiple communities represents a refined community that is valuable to end users. It is worth mentioning that employing a brute-force approach, which involves enumerating all nodes for community prediction and then joining multiple communities to determine the intersection, is an impractical strategy. Therefore, we aim to efficiently identify the intersection without enumerating the dataset. In the literature, some studies have focused on discovering communities that contain multiple query nodes, e.g., [21, 24]. In datasets with overlapping labels, this problem can transition into a single-target or multi-target community search. If the input nodes share one or more community memberships, those communities become the targets. In cases where no common community exists, techniques such as majority voting can be used to identify the target community. As a result, the proposed OCS and OCIS methods can be seamlessly extended to scenarios involving multiple query nodes.

In this paper, the overlapping community search task operates in a semi-supervised framework. Commencing with a graph represented as $G(V, E)$, the model is trained on a small fraction of the dataset (10% or less). Given a graph G, we aim to design a model and a search algorithm to handle both OCS and OCIS. The model should be efficient while capable of handling high-order patterns.

3.2 Graph Convolutional Network

Graph Convolutional Network (GCN) [26] is the most commonly employed variant of GNN, leveraging a low-pass filter (the first-order adjacency matrix) to gather information solely from its neighbors rather than all local nodes. The propagation process is represented as Equation 1:

$$\boldsymbol{H}^{(l+1)} = \sigma(\bar{\boldsymbol{D}}^{-\frac{1}{2}}\bar{\boldsymbol{A}}\bar{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^l\boldsymbol{W}^l), \tag{1}$$

where $\boldsymbol{H}^l$ is the hidden state from the l layer, $\bar{\boldsymbol{D}}$ is the normalized degree matrix which is a diagonal matrix of node degree, $\bar{\boldsymbol{A}}$ is the adjacency matrix with self-loop, $\boldsymbol{W}^l$ is the learnable weight matrix and σ is an activation function. $\bar{\boldsymbol{A}}\boldsymbol{H}^l$ demonstrates how a node aggregates information from its one-hop neighbors. The activation functions add non-linearity between layers and prevent multiple linear functions from collapsing into a single one. Common activation functions include ReLU(·), sigmoid(·), and LeakyReLU(·). Loss is computed by comparing the model output with the ground truth and using backpropagation to update model parameters iteratively.

In contrast to other deep learning models, a deeper GCN does not enhance its expressiveness. With each additional layer, the receptive fields of the GCN expand by one hop. Deeper models lead nodes to aggregate information from the entire graph, diminishing its ability to distinguish nodes, known as oversmoothing [29].

4 SUBSPACE COMMUNITY EMBEDDING: A GENERAL SOLUTION FOR OCS

This section introduces the subspace community embedding technique, a general solution extending any ML-based CS model to OCS. We then propose the SMN in Section 5 as the backbone model to solve existing models' challenges.

Sparsity plays a crucial role in enhancing machine learning models across various applications. It benefits various areas such as subspace clustering [3, 5, 23, 31, 33], sparse training [22, 38], and sparse feature selection [27, 51]. Inspired by this concept, we introduce a subspace embedding technique named Sparse Subspace Filter (SSF). SSF trains node embeddings to align closely with their corresponding community embeddings, minimizing Euclidean and cosine distances within the subspace. This method effectively approximates community representations, enhancing the model's accuracy and relevance in OCS. The sparse subspace filter is initialized as a trainable matrix with (s, c) dimensions, where s is the dimension of the output embeddings, and c denotes the number of communities. In the following, we detail our approach by answering the following questions.

What roles does SSF play in our model? SSF plays two roles in the model, including a filter of the model classifier during the offline training and the basis matrix to guide the community search during the online searching.

Offline training. In the training phase, the model clusters nodes from the same community into a subspace, allowing the trained SSF to function as a basis matrix representing all community subspaces. As illustrated in Figure 3(a), SSF represented by $\boldsymbol{S} \in \mathbb{R}^{s\times c}$ is a sparse

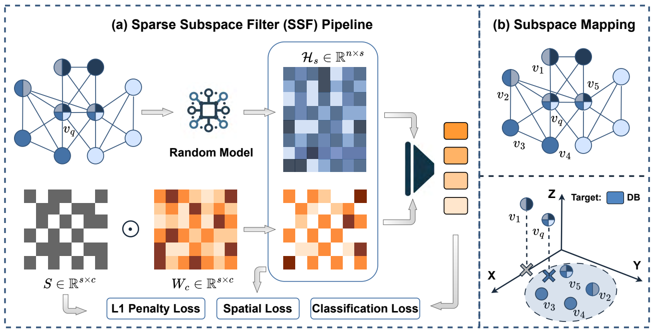

(a) Sparse Subspace Filter (SSF) Pipeline
(b) Subspace Mapping
v_q
Random Model
$\mathcal{H}_s \in \mathbb{R}^{n\times s}$
$S \in \mathbb{R}^{s\times c}$
$W_c \in \mathbb{R}^{s\times c}$
L1 Penalty Loss
Spatial Loss
Classification Loss
v_1 v_2 v_3 v_4 v_5 v_q
Target: DB
Z
X
Y

Figure 3: Subspace community embedding via the sparse subspace filter

matrix with elements drawn from a Bernoulli variable. The black color demonstrates 1 at the underlying position, and the white denotes 0. By performing the element-wise product with the classifier matrix $\boldsymbol{W_c}$, it gates the weight in the classifier, promoting sparsity. Given node embeddings learned by a random model, the gated classifier linearly transforms the node embeddings $\mathcal{H}_s \in \mathbb{R}^{n\times s}$ into the likelihoods of community affiliation. As the columns in the classifier approximate the embeddings for the corresponding community, SSF ensures that each community embedding is related to only a subset of elements in the node embedding. Hence, SSF projects community embeddings into distinct subspaces. This design facilitates overlapping structures as a node embedding with full space covering multiple subspaces simultaneously.

Online searching. During online searching, the underlying columns of SSF can map nodes into the user-selected subspace. Figure 3(b) illustrates how subspace mapping benefits the overlapping community search. For example, if the target domain is the database (DB), node v_1 will be identified as a noise node as it does not have a blue color. However, v_q and v_1 may appear similar in the full space because they share two common labels, making them indistinguishable. By projecting all nodes into the subspace representing DB, v_1 is positioned far from v_q because the elements of its embedding that do not relate to DB are converted to 0, effectively distinguishing it from nodes within that domain. This enhances the model's ability to differentiate nodes based on community relevance.

How are the objective functions designed to train SSF? To make sure the SSF is well-trained to facilitate the overlapping community search, we adopted three objective functions.

L1-penalty term. To induce sparsity while ensuring the objective function is differentiable, we train SSF as the real-valued parameters. We then perform a maximum-likelihood (ML) draw by thresholding the values at 0.5 to sparse the SSF. The L1-penalty term is stated as Equation 2 to ensure the model is in favor of a sparse SSF.

$$\hat{\theta}, \hat{\Phi} = \arg\min_{\theta,\Phi} \left(\ell\left(y \mid \theta, \Phi, y\right) + \lambda \left\|\Phi\right\|\right), \ \ \|\Phi\| = \sum_{i=1}^{s}\sum_{j=1}^{c_i} S_{i,j}, \tag{2}$$

where θ and Φ are the parameters that minimize the loss function, and Φ is the penalty term regulate by ℓ_1 on elements in SSF. The regularization term is scaled by λ to control the level of sparsity. The L1-penalty term ensures that the model is in favor of sparsity, facilitating the model to learn community embeddings falls in different subspaces. This ensures that large communities have a loose constraint in estimating node affiliations, suggesting a higher probability of demonstrating high similarity to node embeddings.

Classification loss. As mentioned in the definition, due to the query in OCS carrying various semantics, the OCS model should first predict the community affiliations, then allow the user to customize their target community. Hence, we adopt a classification loss to supervise the model performance on community prediction. Under overlapping structures, as each node denotes more than one community affiliation, the model tends to suffer the positive-negative imbalance issue. Where most nodes belong to a small fraction of the possible communities, implying the positive samples will be much less than the negative samples. To address this issue, we adopt the ASL loss [34] to assign different exponential decay factors to positive and negative samples. A general form of a binary loss per label, $\mathcal{L}$, is given by Equation 3:

$$\mathcal{L}_c = -y\mathcal{L}_+ - (1-y)\mathcal{L}_-, \tag{3}$$

where $\mathcal{L}_+$ and $\mathcal{L}_-$ are the positive and negative loss parts. Comparatively, ASL loss is defined as Equation 4:

$$\begin{cases} \mathcal{L}_+ = (1-p)^{\gamma_+}\log(p) \\ \mathcal{L}_- = (p_m)^{\gamma_-}\log(1-p_m), \end{cases} \tag{4}$$

where $p_m = \max(p-m, 0)$ is a shifted probability, monitoring p to get $\mathcal{L}_- = 0$ when $p < m$, (γ_+, γ_-) are focusing parameters for positive and negative samples, respectively. ASL balances the contribution from positive and negative samples through a soft threshold (γ_+, γ_-) and a hard threshold (probability margin m). As depicted in Figure 3, the node embeddings $\mathcal{H}_s$ perform a matrix multiplication with the gated classifier $\boldsymbol{W}_c'$ to generate the output logit. The classification loss is calculated by comparing the output logit with the ground truth labels. This loss effectively groups the node embeddings from the same community.

Spatial loss. Furthermore, a spatial loss function is introduced to supervise the subspace mapping, ensuring nodes only fall into the subspaces representing their community affiliations. For nodes belonging to a community, their embeddings should be close to their community in the underlying subspace. Hence, two distance metrics are employed to monitor it, including Euclidean distance and cosine similarity. The loss is primarily generated on the non-zero elements related to each underlying subspace to accommodate overlapping community structures. Therefore, we filter node embeddings by the basis vector of each column in SSF before measuring the distance. Discrepancies against zero-like elements in SSF are not penalized, given their potential contribution to other communities. The sigmoid functions are then applied to the distance and similarity to derive the likelihood of nodes belonging to each community. This output is averaged into the final spatial distance as Equation 5.

$$\boldsymbol{D} = \frac{1}{2}(\sigma(-\mathrm{dist}(\boldsymbol{h}_v, \boldsymbol{W}_c')) + \sigma(\mathrm{sim}(\boldsymbol{h}_v, \boldsymbol{W}_c')), \tag{5}$$

where σ represents a sigmoid function, $\boldsymbol{W}_c'$ represents the gated model classifier, and $\boldsymbol{h}_v \in \mathcal{H}_s$ is the final embeddings. $\mathrm{dist}(\cdot)$ and $\mathrm{sim}(\cdot)$ represent the Euclidean distance and the cosine similarity, respectively. Similar to the classification loss, we compute the ASL

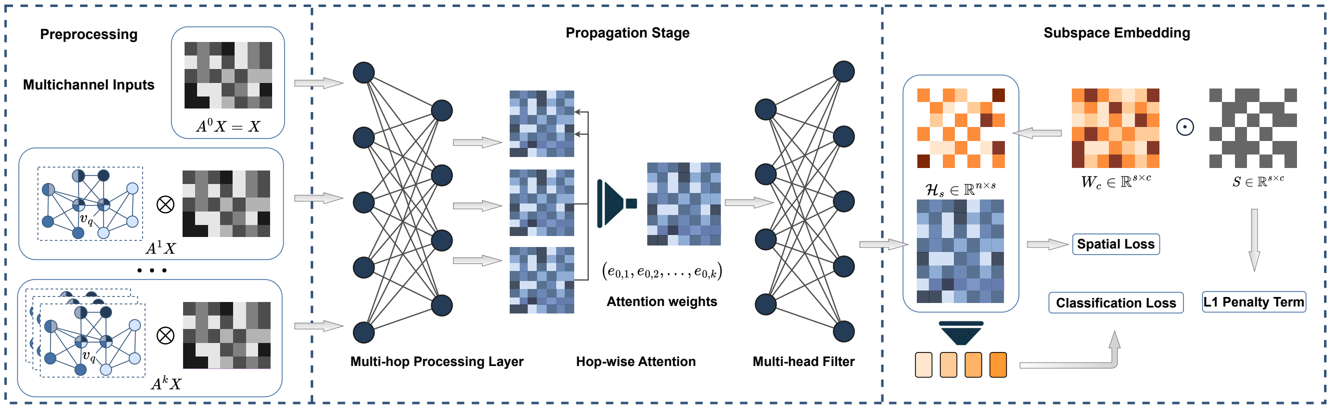


Figure 4: The architecture of SMN

for the spatial loss $\mathcal{L}_s$ against the ground truth as Equation 6:

$$\mathcal{L}_s = \sum_{v=1}^{\eta} ASL(\boldsymbol{d}_v, \boldsymbol{y}_v), \tag{6}$$

Therefore, the final loss function is defined as follows:

$$\mathcal{L} = \frac{1}{2}\left(\frac{\mathcal{L}_c}{\delta_c^2} + \frac{\mathcal{L}_s}{\delta_s^2}\right) + \lambda \left\|\Phi\right\|, \tag{7}$$

where δ is a parameter the model trains to balance the above two loss functions. In the experiments, we observe that this fused loss function stabilizes the model performance.

5 SIMPLIFIED MULTI-HOP ATTENTION NETWORK (SMN)

This section elaborates on the design details of the proposed SMN, which is a lightweight model with large receptive fields. We first introduce the overall framework of SMN to establish a comprehensive understanding as depicted in Figure 4. As the subspace community embedding is already illustrated in Figure 3, in this section, we present the model by mainly focusing on the model preprocessing and propagation phase.

5.1 Framework

Figure 4 presents the framework of SMN, which consists of three main components, including preprocessing, propagation, and subspace community embedding. We prove that the existing widely-used query encoder is not gaining model expressive power but slowing down the training process, detailed analysis disclosed in Section 7. Hence, we removed the query encoder for model efficiency. In addition, we adopted a simplified framework to further improve the training speeds. This framework removes activation functions between layers and aggregates multi-hop neighborhood messages during preprocessing instead.

In the preprocessing stage, SMN generates multichannel inputs by stacking messages from different hops. The k-th channel represents the feature matrix with $(k-1)$ hops awareness. Aggregating neighborhood information during preprocessing eliminates the need for expensive message-passing during the model propagation. Hence, the model training speeds are further accelerated.

The propagation stage consists of three layers: a multi-hop processing layer, a hop-wise attention layer, and a multi-head filter layer. The multi-hop processing layer inputs the original features from each hop to linearly transform the multi-hop messages. The messages are then fused into single-channel messages through a hop-wise attention layer. The resulting outputs are further transformed through a multi-head filter layer, yielding the final embeddings. The hop-wise attention mechanism assigns decaying weights to messages from various hops based on their contributions. This design effectively addresses the oversmoothing issue, enhancing model high-order awareness to facilitate overlapping community search. The final embeddings are then fed into SSF, learning subspace community embedding for OCS.

5.2 SMN: Preprocessing and Propagation

Preprocessing. The preprocessing stage can be split into aggregation and normalization.

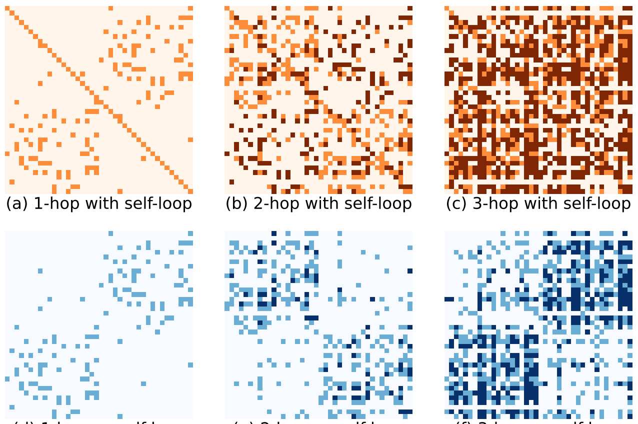
Aggregation. Inspired by the works [19, 47, 49], SMN removes the non-linear activation functions during aggregation to improve the model training speed. As proved by Wei et al. [48], linear propagation performs similarly to non-linear propagation, especially when graph structures are more informative compared to node attributes. A two-layer GCN can be represented as Equation 8:

$$Z = \mathrm{softmax}(\hat{\boldsymbol{A}} \times \mathrm{ReLU}(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}^{(0)})\boldsymbol{W}^{(1)}), \tag{8}$$

where $\boldsymbol{Z}$ is the final output, softmax is a classifier that maps the probability of nodes belonging to different classes. ReLU$(\cdot)$ is the activation function to provide nonlinearity to the model. $\hat{\boldsymbol{A}}$ denotes the degree normalized adjacent matrix, $\boldsymbol{X}$ is a matrix of node features and $\boldsymbol{W}$ is a learnable matrix. $\boldsymbol{W}^0\boldsymbol{W}^1$ represents the weight for different layers of the networks. By removing the activation functions, SMN can be represented as Equation 9:

$$\begin{aligned} Z &= \mathrm{softmax}(\hat{\boldsymbol{A}} \times (\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}^{(0)})) \\ &= \mathrm{softmax}(\hat{\boldsymbol{A}}^2\boldsymbol{X}\boldsymbol{W}^{(0)}). \end{aligned} \tag{9}$$

Since the computation of $\hat{\boldsymbol{A}}^2\boldsymbol{X}$ is equal to a preprocessing step, the level of the parameter is the same as a logistic regression model. However, this simplified framework faces an obvious limitation: by aggregating the neighborhood message, the node features will


(a) 1-hop with self-loop (b) 2-hop with self-loop (c) 3-hop with self-loop

(d) 1-hop no self-loop (e) 2-hop no self-loop (f) 3-hop no self-loop

Figure 5: Self-loop oversmooth messages received

quickly become indistinguishable. Therefore, this limits the model to a relatively low-hop awareness and harms OCS.

Instead of directly using $\hat{\boldsymbol{A}}^k\boldsymbol{X}$ as input, SMN iteratively stacks adjacency matrices from different hops $\hat{\boldsymbol{A}}^0, \hat{\boldsymbol{A}}^1, ...\hat{\boldsymbol{A}}^k$, and generates multichannel inputs by assigning node features such as $\boldsymbol{X}, \hat{\boldsymbol{A}}\boldsymbol{X}, ...\hat{\boldsymbol{A}}^k\boldsymbol{X}$. The k-th channel represents the node feature matrix with a $k-1$ hop receptive field. By aggregating neighborhood information during preprocessing, SMN eliminates the necessity of a GCN layer but employs a fully connected layer instead.

Normalization. In degree normalization, the adjacency matrix of GCN is normalized as $\hat{\boldsymbol{A}} = \bar{\boldsymbol{D}}^{-\frac{1}{2}}\bar{\boldsymbol{A}}\bar{\boldsymbol{D}}^{-\frac{1}{2}}$, where $\bar{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ and $\boldsymbol{I}$ is the identity matrix, representing each node in $\boldsymbol{A}$ to add a self-loop by $\boldsymbol{A} + \boldsymbol{I}$ before normalization. The self-loop avoids the loss of self-features during aggregation. In contrast, SMN specifically removes the self-loop for two reasons: First, SMN takes the input of $\tilde{\boldsymbol{A}}^0\boldsymbol{X}, \tilde{\boldsymbol{A}}^1\boldsymbol{X}, ...\tilde{\boldsymbol{A}}^k\boldsymbol{X}$, where $\tilde{\boldsymbol{A}}^0\boldsymbol{X} = \boldsymbol{X}$ is the initial features matrix which prevents the loss of the self-features. Furthermore, removing the self-loop reduces redundancy during message passing and further differentiates messages collected from each hop. Thus, degree normalization in SMN is depicted in Equation 10:

$$\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}, \tag{10}$$

where $\boldsymbol{A}$ and $\boldsymbol{D}$ represent the adjacency and the degree matrices without self-loop.

Figure 5 illustrates the comparison of adjacency matrices with and without self-loops. It shows that adding self-loops leads to a notable acceleration in graph exploration, causing oversmoothing within three hops. Adversely, removing self-loops leads to a better contrast across the adjacency matrices among various hops. It can be seen that the Figure 5(d) (1-hop no self-loops matrix) primarily focuses on direct neighbors, presented as the matrix's top-right and bottom-left corners. In contrast, the 2-hop matrix (Figure 5(e) emphasizes neighbors with a 2-hop distance (top-left and bottom-right corners), ignoring the 1-hop neighborhood. By comparing the Figure 5(c) and 5(f), removing self-loops effectively slows down the oversmoothing progress while enabling the proposed attention mechanism to capture unique patterns from various hops.

Propagation: hop-wise attention. The hop-wise multi-head attention mechanism regulates the aggregation of messages from various hops, enabling the model to capture higher-order patterns

Algorithm 1: Preprocessing and SMN propagation

```
Input  : Feature matrix $X$, the adjacency matrix $\tilde{A}$, the number of hops $k$,
         the sparsity rate $r$
Output : Model output $O$, final embeddings $\mathcal{H}_s$, learned sparse subspace
         filter $S$
1  $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$;
2  $\mathcal{H} = \{\tilde{A}^0X, \tilde{A}^1X, ..., \tilde{A}^{(k-1)}X\}$;
3  for each $H_i \in \mathcal{H}$ do
4      $H_i = \sigma(W^l H_i)$;
5      $\alpha_i = \text{softmax}(a(H_0, H_i))$;
6  end for
7  $\mathcal{H} = \text{AGG}(\alpha_i H_i)$, for $i = 0, ..., k-1$;
8  $\mathcal{H}_s = \sigma(W^r \mathcal{H})$;
9  Initialize the subspace filter $S$;
10 $W_c' = \text{ApplySparsity}(S, W_c)$;
11 $O = \mathcal{H}_s \times W_c'$;
12 return $O, \mathcal{H}_s, S$;
```

while mitigating the oversmoothing effect. SMN first applies a multi-hop processing layer to transform the initial features linearly to obtain sufficient expressive power. Here, the weight matrix $\boldsymbol{W}^l$ is shared across nodes and hops. We then perform self-attention on the hidden state to compute the attention coefficients as Equation 11:

$$\begin{aligned} \boldsymbol{e_i} &= a(\boldsymbol{W}^l\boldsymbol{H}_0, \boldsymbol{W}^l\boldsymbol{H}_i), \quad \forall i \in [0..k] \\ &= (\overrightarrow{\boldsymbol{a}}^T\boldsymbol{W}^l\boldsymbol{H}_0 + \overrightarrow{\boldsymbol{a}}^T\boldsymbol{W}^l\boldsymbol{H}_i), \end{aligned} \tag{11}$$

where $\boldsymbol{e_i}$ indicates the importance of $\boldsymbol{H}_i$ the ith-hop features toward $\boldsymbol{H}_0$ the zero-hop (self-features matrix). $\overrightarrow{\boldsymbol{a}}$ is a shared attention mechanism $\overrightarrow{\boldsymbol{a}} \in \mathbb{R}^{d'}$, and k is the number of hops. To fuse the message from different hops, the coefficients are first activated by a LeakyReLU [41], which improves stability by allowing a small gradient for negative inputs, and then normalized by the softmax as Equation 12. The obtained final weights $\boldsymbol{\alpha}$ weighted sum the multi-hop feature matrices into a single channel.

$$\boldsymbol{\alpha}_i = \frac{\exp\left(\text{LeakyReLU}\left(\boldsymbol{e_i}\right)\right)}{\sum_{j\in[0..k]}\exp\left(\text{LeakyReLU}\left(\boldsymbol{e_j}\right)\right)}. \tag{12}$$

To further improve the performance, we observe that multi-head attention is beneficial in stabilizing the performance. Similar to the graph attention networks [41], multiple independent attention mechanisms are applied to the hidden state, and the output of each head is further concatenated into the final output. The model uses the multi-head filter $\boldsymbol{W}^r$ to fuse the output from different heads into final embeddings as Equation 13:

$$\mathcal{H}_s = \sigma\left(\boldsymbol{W}^r\left(\Vert_{i=1}^{I}\sum_{k=0}^{K}\boldsymbol{\alpha}_k^i\boldsymbol{W}^l\boldsymbol{H}_k\right)\right), \tag{13}$$

where I is the number of heads and K is the number of hops. The dimension of the final output $\mathcal{H}_s$ is a hyper-parameter that matches the dimensions of the subspace community embeddings.

The algorithm for SMN propagation is presented in algorithm 1. Lines 1-2 represent the preprocessing stage, stacking aggregated features from different hops. Lines 3-8 describe the model propagation stage. The preprocessed features are linearly transformed by $\boldsymbol{W}^l$ and then fused by the weight from hop-wise attention. The fused hidden state is then transformed by $\boldsymbol{W}^r$. Lines 9-11 describe the subspace embeddings. This hop-wise attention mechanism enhances

SMN's flexibility by attending to broader receptive fields, capturing the unique graph structure across different real-life datasets.

6 ONLINE SEARCH PHASE

This section provides the design details for extending the current ML-based model's online search phase [24, 30] to OCS. We then analyze the feasibility of applying the proposed method to OCIS to identify the intersection of multiple targets effectively.

6.1 Overlapping Communities Search (OCS)

Leveraging the subspace community embedding technique, we first extend the naive top-k similarity search to an OCS named Sub-Topk. Considering Sub-Topk's limitations, we proposed a spatial-aware algorithm called subspace cohesive community search (Sub-CS).

Sub-Topk. We first propose Lemma 6.1 states that the classifier $\boldsymbol{W}_c'$ gated by SSF approximates community embeddings in overlapping community structures. Based on this result, we initiate a similarity-based approach called Sub-Topk to identify a query-dependent community. The algorithm takes the query nodes and the test set as input, mapping all nodes to the target subspace by performing an element-wised dot product against the basis vector.

Lemma 6.1 (Sparse Classifier Approximates Global Centroid of Communities). *Given a set $X_j = \{x_1, x_2, ..., x_{|N_j|}\}$ of node embeddings in $\mathbb{R}^s$ that belong to community j, and a classifier vector $w_j \in \boldsymbol{W}_c'$. Through the learning process, w_j will converges to the centroid μ_j of community j defined by:*

$$\mu_j = \frac{1}{|N_j|} \sum_{x_i \in X_j} x_i.$$

Proof. The updated rule for using gradient descent is given by:

$$w_j \leftarrow w_j + \alpha \sum_{i \in N_j} x_i \cdot (y_{ij} - \sigma(x_i^\top w_j)),$$

where y_{ij} is the indicator function, σ is the activation function, and α is the learning rate. As the model learns, the predicted probabilities $\sigma(x_i^\top w_j)$ approach the true class labels y_{ij}, reducing the term $y_{ij} - \sigma(x_i^\top w_j)$ to a small error ϵ_{ij} near zero. $\Delta w_j = \alpha \sum_{i \in N_j} x_i \epsilon_{ij}$, with ϵ_{ij} trending towards zero as classification accuracy improves. Assuming ϵ_{ij} becomes negligible, the updates to w_j become smaller, stabilizing w_j around a vector that maximizes the sum of projections of x_i on w_j. This stabilization point is given by:

$$\lim_{\epsilon_{ij} \to 0} w_j \approx \frac{\sum_{i \in N_j} x_i}{|N_j|} \approx \mu_j.$$

□

A well-trained SSF functions as a subspace community embedding, effectively filtering out noisy nodes. This subspace mapping segregates the target community from overlapping communities.

Example 6.1. *For a node v with a feature vector* $\mathbf{x} = [0.82, 0.11, -0.69, -1.3, 0.03]$, *there are two distinct communities with basis vectors e.g.,* $S_1 = [1, 0, 1, 1, 0]$, $S_2 = [0, 1, 0, 0, 1]$. *The projections are computed as follows:*

$$x_1' = x \odot S_1 = [0.82, 0, -0.69, -1.3, 0],$$

$$x_2' = x \odot S_2 = [0, 0.11, 0, 0, 0.03].$$

The vector x_1' is likely to demonstrate higher similarity with nodes in the subspace defined by S_1, compared to a lower probability with nodes in subspace S_2. Therefore, the subspace mapping effectively filters out irreverent nodes during the search.

Algorithm 2: Cohesive community search (Sub-CS)

```
Input   :Graph $G$, Query $v_q$, final embeddings $\mathcal{H}_s$, learned sparse
         subspace filter matrix $S$, community size $k$, similarity threshold $l$
Output  :Community $C_q$
1  $\mathcal{H}_s$ = SubspaceMapping($\mathcal{H}_s, S_q$);
2  $C_q = \{q\}$;
3  for each $v$ encountered in $G$ sorted by similarity against $q$ do
4      Add $v$ to $C_q$ if $|C_q| < k$ ;
5      $\bar{x}$ = mean($\mathcal{H}_s[i]$), $i \in C_q$ ;
6      $P$ = sim($C_q, \bar{x}$);
7      Find a node $v' \in C_q$ with $P[v']$ smallest in $C_q$;
8      if $v' = q$ then break;
9      if sim($v, q$) <= $l$ then break;
10     $p_v$ = sim($v, \bar{x}$);
11     if $p_v > P[v']$ then
12         $C_q$.remove($v'$); $C_q$.add($v$);
13         $\bar{x}$ = mean($\mathcal{H}_s[i]$), $i \in C_q$ ;
14     end if
15 end for
16 return $C_q$;
```

Sub-CS. While Sub-Topk effectively segregates unrelated nodes, it operates under an unrealistic assumption that the query node is always at the centroid of the community. Inspired by the idea of spatial-aware community search [11, 15], we further introduce a subspace-aware community search (Sub-CS) to identify a densely interrelated community in the latent space, allowing a shift in the community centroid. The algorithm aims to locate a community with a small "radius" in the subspace. Subspace cohesiveness implies that the identified community should minimize the community radius in the latent subspace, with distance measured by cosine similarity. Sub-CS explores top-k nodes demonstrating the highest similarity to the query as the initial community; by traversing through the dataset following a descending order of node similarity, Sub-CS updates the community to maximize the group similarity. Let v_q denote the query node, and $\mathbf{C_0}$ represent the initial community. Compute the centroid $\bar{x}$ of $\mathbf{C_0}$ as the mean of the nodes embeddings in $\mathbf{C_0}$, i.e., $\bar{x} = \frac{1}{|C_0|} \sum_{v \in C_0} x$, where x is the embedding vector of node v. Subsequently, we evaluate each node v in C_0 against $\bar{x}$ using cosine similarity. Replace the least similar node v' with a new encountered node v_i if v_i exhibits higher similarity as Equation 14:

$$C_{i+1} = \begin{cases} C_i \setminus \{v'\} \cup \{v_i\} & \text{if } \cos(x_i, \bar{x}) > \cos(x', \bar{x}), \\ C_i & \text{otherwise.} \end{cases} \tag{14}$$

This process continues and recomputes the centroid if the community is updated. The algorithm terminates if the query node becomes the least similar node or the early stop condition is reached. The early stop controls the node similarity against the query, preventing the final community from including dissimilar nodes. In the experiment, we set the threshold at $2 \times k$ nodes, which means that the algorithm will only consider nodes with top-$2k$ similarity. The details are illustrated in algorithm 2. The Lemma 6.2 states that the community results in a decreasing radius by interactively removing the least similar node measured by cosine similarity.

Lemma 6.2 (The Smallest Radius in Embedding Space). *Let $C_i = \{x_1, x_2, \ldots, x_m\}$ be a set of points in $\mathbb{R}^s$ is the embeddings of nodes in a identified community. Let $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$ be the centroid of these nodes. Assume $x' \in C_i$ is the node with the minimum cosine similarity to $\bar{x}$, and $x_{m+1} \notin C_i$, having $\cos(\bar{x}, x_{m+1}) > \cos(\bar{x}, x')$. When x' is replaced by x_{m+1}, resulting in a new centroid $\bar{x}'$, then:*

$$\sum_{i=1}^{m} \cos(\bar{x}', x_i) > \sum_{i=1}^{m} \cos(\bar{x}, x_i).$$

Proof. Given $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$ is the centroid, the new centroid after replacing x' with x_{m+1} is: $\bar{x}' = \bar{x} - \frac{1}{m}x' + \frac{1}{m}x_{m+1}$. Define $\bar{x}^* = \bar{x} - \frac{1}{m}x'$ as the centroid after removing x'. Given $\cos(\bar{x}, x_{m+1}) > \cos(\bar{x}, x')$, we have: $\cos(\bar{x}^* + \frac{1}{m}x', x_{m+1}) > \cos(\bar{x}^* + \frac{1}{m}x', x')$. Since $\cos(x', x') = 1$ and for any $x_{m+1} \neq x'$, $\cos(x', x_{m+1}) < 1$, it follows that: $\cos(\bar{x}^*, x_{m+1}) > \cos(\bar{x}^*, x')$. As $\bar{x}^*$ represents all other nodes in the community excluding x', the similarity of the new centroid $\bar{x}'$ with each x_i increases:

$$\cos(\bar{x}', x_i) = \frac{(\bar{x} - \frac{1}{m}x' + \frac{1}{m}x_{m+1}) \cdot x_i}{\|\bar{x} - \frac{1}{m}x' + \frac{1}{m}x_{m+1}\|\|x_i\|}.$$

Summing these, $\sum_{i=1}^{m} \cos(\bar{x}', x_i)$ is greater than $\sum_{i=1}^{m} \cos(\bar{x}, x_i)$, proving Lemma 6.2. □

In OCS, cosine similarity, despite not being a metric space, offers key advantages for our task. Since nodes often belong to multiple communities, their feature vectors may have smaller values, and the magnitude can vary significantly between popular (multi-community) and less popular nodes. By emphasizing directional alignment rather than magnitude, cosine similarity allows us to focus on the structural similarity of nodes within a target community. This approach effectively handles variations in node popularity, enabling better identification of nodes aligned with the target community regardless of their overall influence in the network.

6.2 Overlapping Communities Intersection Search (OCIS)

In OCIS, SMN provides enhanced flexibility to end users by allowing the selection of multiple communities as the target and returning only to their intersection. The brute-force approach identifies all target communities and determines their intersection by examining common nodes. However, this method leads to high computation overhead, as it requires enumerating the entire dataset for community prediction, followed by intersection-finding operations. This process is equivalent to solving a community detection problem, which becomes time-consuming when handling large graphs, especially when the target result involves only a small subset of nodes. Leveraging subspace embedding techniques, SMN efficiently identifies the intersection while avoiding computational wastage. The rationale is that nodes in the intersection set should exhibit closer relationships with all community embeddings involved. Lemma 6.3 states that if nodes demonstrate high similarity in two subspaces, they will also be similar in their unioned space.

Lemma 6.3 (Cosine Similarity Preserved in Unioned Subspace). *Given k communities, each represented by a distinct subspace S_i where $i \in \{1, 2, \ldots, k\}$. Define the unioned subspace U as $U = \bigcup_{i=1}^{k} S_i$. If two nodes x_1 and x_2 demonstrate high cosine similarity in each distinct subspace S_i, then x_1 and x_2 will also demonstrate substantial cosine similarity in the unioned subspace U.*

Proof. Let $x_1^{(i)}$ and $x_2^{(i)}$ be the representations of x_1 and x_2 in the subspace S_i. Since x_1 and x_2 demonstrate high cosine similarity in subspaces S_i, we have: $\cos(x_1^{(i)}, x_2^{(i)}) = \frac{x_1^{(i)} \cdot x_2^{(i)}}{\|x_1^{(i)}\|\|x_2^{(i)}\|} \approx 1 \quad \forall i \in \{1, 2, \ldots, k\}$. Since $x_1^U \cdot x_2^U$ is the sum of the dot products in each subspace S_i: $x_1^U \cdot x_2^U = \sum_{i=1}^{k} x_1^{(i)} \cdot x_2^{(i)}$. Given that $\cos(x_1^{(i)}, x_2^{(i)}) \approx 1$, we have: $x_1^{(i)} \cdot x_2^{(i)} \approx \|x_1^{(i)}\|\|x_2^{(i)}\| \quad \forall i \in \{1, 2, \ldots, k\}$. Therefore:

$$x_1^U \cdot x_2^U \approx \sum_{i=1}^{k} \|x_1^{(i)}\|\|x_2^{(i)}\| \approx \|x_1^U\|\|x_2^U\|.$$

This shows that if x_1 and x_2 demonstrate high cosine similarity in each distinct subspace S_i, they will also demonstrate substantial high cosine similarity in the unioned subspace U. □

7 THEORETICAL ANALYSIS

Analysis of the query-graph encoder. We first present a theoretical analysis explaining why the widely used query-graph encoder frameworks in ML-based approaches [24, 30] cause computational overhead leading to $O(|V|^2)$ training time complexity.

Lemma 7.1 (Time Complexity of the Query-Graph Encoder Framework). *Fusing the hidden states of both encoders' output at each layer will lead to $O(|V|^2)$ time complexity in model training.*

Proof. Given that the query encoder processes each node individually, similar to Stochastic Gradient Descent (SGD), the training will involve $|V|$ batches, each sized as 1. The time complexity is $O(1)$ for each batch, and for the entire training set, it is $|V| \times O(1) = O(|V|)$. The graph encoder processes the full graph each time with a single batch sized $|V|$. The time complexity is $O(|V|)$ for each batch, and for the entire training set, it remains $O(|V|) = 1 \times O(|V|)$. When the two encoders are fused, there are $|V|$ batches, each sized as $(1, |V|)$. This means that for each node processed by the query encoder, the graph encoder processes the entire graph. Consequently, the total time complexity for each batch increases to $|V|$, which results in $|V| \times O(|V|) = O(|V|^2)$. □

Hence, the query-graph encoder framework is the primary reason for the slow training issues in existing CS models. Moreover, while this framework is claimed to capture both local and global information, it primarily affects the gradient descent optimization. Ultimately, this impact averages out over the training, behaving similarly to standard SGD as shown in Lemma 7.2.

Lemma 7.2 (The optimization is Equivalence to SGD). *Let θ be the parameter vector of a neural network trained using a query-graph encoder where $Q(v_i)$ adjusts θ locally for each node v_i and $G(G)$ provides global adjustments based on the entire graph G. The cumulative effect of these adjustments over multiple training epochs is equivalent to the effect of Stochastic Gradient Descent (SGD) on θ.*

Proof. The local adjustments by $Q(v_i)$ for each node and the global adjustments by $G(G)$ can be formally expressed as:

$$\theta \leftarrow \theta - \eta \left(\nabla L_{\text{local}}(Q(v_i), \theta) + \nabla L_{\text{global}}(G(G), \theta) \right),$$

where η is the learning rate, and ∇L_{local} and ∇L_{global} are the gradients of the loss functions localized to $Q(v_i)$ and globalized to $G(G)$, respectively. In traditional SGD, parameter updates are influenced by the gradient of the loss function evaluated at different subsets of the data. Over many iterations, this results in:

$$\theta \leftarrow \theta - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla L_{\text{local}}(Q(v_i), \theta),$$

where n is the total number of nodes. Given the high frequency of updates involving every node v_i and the entire graph G, the effects of $Q(v_i)$ and $G(G)$ fusion leading each parameter update by $Q(v_i)$ is averaged with updates induced by $G(G)$:

$$\theta \leftarrow \theta - \eta \left(\frac{1}{n} \sum_{i=1}^{n} \nabla L_{\text{local}}(Q(v_i), \theta) + \nabla L_{\text{global}}(G(G), \theta) \right).$$

This aligns with the principle of SGD, which states that the aggregate update is the average of the updates across all data points. □

SMN time complexity analysis. We provide detailed considerations for both model training and community identification, addressing preprocessing and query time complexities, respectively. The feature processing adopts k-hop operations in the preprocessing stage, contributing $O(|V|^3 \times k)$. The subsequent multi-hop processing layer involves $O(|V| \times d \times h)$, where d is the initial feature dimensions, and h is the hidden dimensions. Notably, for vanilla GNN models, the aggregation process with $O(|V|^3 \times k)$ happens during the model training. This slows down the model training speed due to repetitive propagation and backpropagation operations. Comparatively, the above preprocessing only operates once before training, avoiding expensive overhead during the training. The multi-hop attention introduces $O(|V| \times i \times h)$ complexity, where i is the number of heads. Hop-wise addition and weighted average fusion will take $O(|V|)$. As these operations will be run for k time, the total time complexity for the multi-hop processing and multi-hop attention layers is $O(k \times |V| \times h \times (d+i))$. The multi-head filter and SSF transformation will take $O(|V| \times h \times s)$ and $O(|V| \times s \times c)$. Where s represents the dimensions of SSF and c is the number of communities. The model is trained by t epochs. Therefore, the total time complexity for SMN training is $O(|V|^3 \times k + (|V| \times t \times (h \times k \times (d+i) + s \times (h+c))))$.

For Sub-Topk, applying the target SSF to map node features and computing cosine similarity against the query node will take $O(|V| \times h)$. To get the top k similarity, will take $O(|V| \times log(k))$, where k is the community size. Therefore, the total time complexity for Sub-Topk will be $O(|V| \times (h + log(k)))$.

For Sub-CS, applying the target SSF to map node features will take $O(|V| \times h)$. To get the top k similarity, will take $O(|V| \times log(k))$, where k is the community size. To update the centroid will take $O(k^2 \times h)$. Therefore, the total time complexity will be $O((k^2 \times h) + |V| \times (h + log(k)))$.

8 EXPERIMENTS

In this section, we conduct experiments on 13 datasets to demonstrate the effectiveness and efficiency of the proposed techniques from 5 perspectives. We first report the model performance on OCS and OCIS to demonstrate the effectiveness in overlapping community structures. Then, we illustrate the efficiency comparison to show the superiority of SMN in both model training and query processing. Thirdly, we compare the model performance in disjoint datasets to show that even though the SMN and SSF are primarily built for OCS, they can also effectively handle disjoint community search. Subsequently, we conduct the ablation study to analyze the contribution of each building block. Finally, we provide hyper-parameters analysis to reveal insights into model parameters.

Table 1: Dataset statistics

Dataset		# Nodes	# Edges	# Com	# Feat	OR	MLA
Overlap	FB-0	185	645	3	224	0.188	3
	FB-107	418	4,815	4	576	0.02	2
	FB-348	207	2,716	4	161	0.744	4
	FB-414	108	954	2	105	0.065	2
	FB-686	159	1,607	6	63	0.698	6
	Chemistry	35,409	157,358	14	4,877	0.25	13
	CS	21,957	96,750	18	7,793	0.275	13
	Engineering	14,927	49,305	16	4,839	0.272	12
	Medicine	63,282	810,314	17	5,538	0.365	16
Disjoint	Cora	2,708	5,429	7	1,433	-	-
	Citeseer	3,312	4,732	6	3,703	-	-
	Pubmed	19,717	44,338	3	500	-	-
	Reddit	232,965	114M	41	602	-	-

8.1 Experimental Setup

Datasets. We use 13 datasets to evaluate the performance of SMN, including 9 datasets with overlapping community structures and 4 datasets demonstrating disjoint structures. Datasets statistics are reported in Table 1. Facebook [28] is a social network dataset that contains five ego networks. Chemistry, Computer Science, Medicine, and Engineering are co-authorship networks constructed using data from the Microsoft Academic Graph (MAG) [1]. Cora, Citeseer, and Pubmed are citation networks with details disclosed on Relational Dataset Repository[2]. Reddit [17] is an online forum where nodes are posts, and edges are comments from the same user. Overlap Ratio ($OR = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(|L_i| > 1)$) and Max Label Affiliations ($MLA = \max(|L_i|)$) measure how overlap the dataset is.

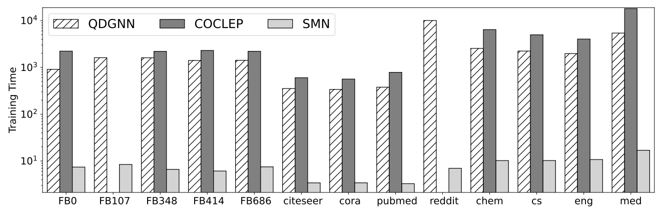
Data splitting. By following the popular semi-supervised settings [49], SMN uses 10% or less labeled data during the training to mitigate the human effort on labeling. In citation networks, a standard splitting is applied [19], utilizing only 20 samples from each community, accounting for less than 2% of the total data. For the Facebook, Reddit, and MAG datasets, the splitting ratio is 10:10:80. The training set is exclusively used to compute the loss and update model parameters. During validation, the parameters are frozen, and the test set remains untouched to prevent information leakage.

Baseline models. We compare the performance of SMN against three algorithm-based methods (k-clique [4], CTC [21], and k-core based [37]) and three SOTA GNN-based models (ICS-GNN [14], QDGNN [24], and COCLEP [30]). GNN-based models are all primarily focused on disjoint community structure. ICS-GNN shares the same configuration as the proposed SMN, identifying a k-sized community. QDGNN and COCLEP identify communities by training a threshold to measure GNN score and similarity. We extend their configuration to a k-sized community search by selecting

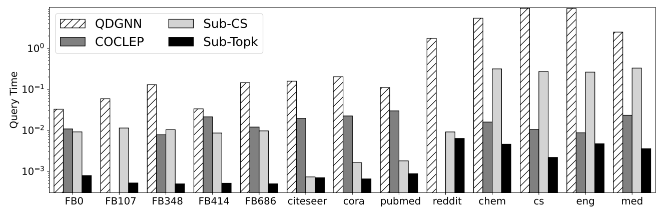
[1]https://www.microsoft.com/en-us/research/project/open-academic-graph/
[2]https://relational.fit.cvut.cz/

Table 2: SMN performance in overlapping community search

Task		Overlapping Community Search, OCS								Overlapping Communities Intersection Search, OCIS								
Metric	Model	k-clque	CTC	k-core	ICS GNN	QD GNN	COC LEP	SMN Topk	SMN CS	k-clque	CTC	k-core	ICS GNN	QD GNN	COC LEP	SMN Topk	SMN CS	Ave+
F1	FB0	0.2478	0.2588	0.2423	0.7058	0.6710	0.2424	0.7427	**0.7630**	0.0572	0.0622	0.0551	0.6122	0.5982	0.6667	0.6547	**0.7147**	35%
	FB107	0.2781	0.3024	0.2537	0.6835	0.6361	-	0.9035	**0.9103**	0.0712	0.0829	0.0609	0.5127	0.5760	-	**0.7520**	0.6520	46%
	FB348	0.1543	0.1366	0.1443	0.8041	0.7338	0.6907	**0.8517**	0.7913	0.0916	0.0949	0.0840	0.7508	0.7316	0.6822	**0.8114**	0.8031	39%
	FB414	0.2882	0.3119	0.2718	0.7941	0.6923	0.7286	0.8745	**0.9006**	0.0798	0.0907	0.0681	0.4107	0.4813	0.2080	0.7493	**0.7533**	45%
	FB686	0.0947	0.0881	0.1013	0.6366	0.6006	0.6512	0.6776	**0.7075**	0.0691	0.0825	0.0615	0.4077	0.4351	0.4201	0.4958	**0.5966**	32%
	ENG	0.0471	0.0529	0.0553	0.6680	0.7422	0.1530	**0.8172**	0.7618	0.0471	0.0529	0.0553	0.6406	0.6792	0.1659	**0.8096**	0.7973	52%
	CS	0.0395	0.0433	0.0408	0.6187	0.5878	0.1400	**0.8301**	0.8242	0.0395	0.0433	0.0408	0.6426	0.6472	0.1507	0.7383	**0.7504**	53%
	CHEM	0.0594	0.0615	0.0623	0.5732	0.6151	0.1812	**0.8585**	0.8487	0.0594	0.0615	0.0623	0.6047	0.6940	0.2199	0.8734	**0.8758**	59%
	MED	-	0.0503	0.0622	0.6630	0.5704	0.1628	0.8416	**0.8540**	-	0.0503	0.0622	0.6760	0.6927	0.1651	0.8405	**0.8514**	53%
JAC	FB0	0.1972	0.2115	0.1903	0.5446	0.5049	0.1379	0.5907	**0.6168**	0.0559	0.0609	0.0538	0.6022	0.5811	0.5172	0.6500	**0.7080**	34%
	FB107	0.2386	0.2768	0.2048	0.5192	0.4664	-	0.8783	**0.8913**	0.0709	0.0827	0.0606	0.5113	0.5760	-	**0.7520**	0.6520	49%
	FB348	0.1116	0.0940	0.1128	0.6724	0.5796	0.5275	**0.7417**	0.6547	0.0874	0.0924	0.0771	0.6649	0.6447	0.5460	**0.7233**	0.7157	36%
	FB414	0.2538	0.2931	0.2294	0.6585	0.5294	0.5731	0.7769	**0.8191**	0.0795	0.0903	0.0673	0.3987	0.4680	0.2080	0.7380	**0.7420**	45%
	FB686	0.0661	0.0599	0.0728	0.4669	0.4292	0.4828	0.5124	**0.5474**	0.0641	0.0796	0.0554	0.3623	0.4002	0.2793	0.4645	**0.5597**	29%
	ENG	0.0260	0.0296	0.0311	0.5015	0.5901	0.0828	**0.6908**	0.6152	0.0260	0.0296	0.0311	0.6259	0.6634	0.0917	**0.7799**	0.7659	49%
	CS	0.0224	0.0249	0.0233	0.4479	0.4162	0.0753	**0.7096**	0.7009	0.0224	0.0249	0.0233	0.6124	0.6244	0.0839	0.7101	**0.7206**	51%
	CHEM	0.0349	0.0363	0.0369	0.4017	0.4442	0.0996	**0.7522**	0.7372	0.0349	0.0363	0.0369	0.5744	0.6728	0.1298	**0.8403**	0.8392	58%
	MED	-	0.0288	0.0368	0.4959	0.3990	0.0886	0.7266	**0.7453**	-	0.0288	0.0368	0.6404	0.6472	0.0933	0.7946	**0.8054**	52%
NMI	FB0	0.1788	0.1245	0.2069	0.1535	0.2007	0.1029	**0.3182**	0.2905	0.1788	0.1245	0.2069	0.2117	0.2021	0.1673	0.5212	**0.5418**	25%
	FB107	0.3790	0.5479	0.2054	0.1590	0.2794	-	**0.6176**	0.5937	0.3790	0.5479	0.2054	0.1554	0.2043	-	**0.6395**	0.6197	31%
	FB348	0.3338	0.4700	0.3321	0.4626	0.4155	0.2345	0.5301	**0.6550**	0.3338	0.3380	0.3321	0.2023	0.1760	0.0771	**0.3829**	0.3582	17%
	FB414	0.3695	0.4281	0.4250	0.4449	0.3914	0.3189	0.5669	**0.6186**	0.3695	0.4281	0.4250	0.3529	0.4286	0.1375	0.6318	**0.6325**	24%
	FB686	0.2862	0.2790	0.2225	0.2047	0.2864	0.1773	**0.4040**	0.3777	0.2862	0.2790	0.2225	0.2474	0.2662	0.2608	**0.4723**	0.4495	17%
	ENG	0.0424	0.0545	0.0687	0.3201	0.4550	0.0325	**0.5803**	0.4810	0.0424	0.0545	0.0687	0.3094	0.4986	0.0333	**0.7696**	0.7590	48%
	CS	-	0.0377	-	0.2936	0.2983	0.0097	0.5954	**0.6033**	-	0.0377	-	0.2985	0.4734	0.0047	0.6891	**0.7043**	47%
	CHEM	0.0393	0.0396	0.0411	0.2745	0.2961	0.0297	**0.6546**	0.6405	0.0393	0.0396	0.0411	0.2636	0.4930	0.0107	**0.7028**	0.6896	54%
	MED	-	0.0556	0.0430	0.3916	0.2744	0.0746	0.6419	**0.6726**	-	0.0556	0.0430	0.3806	0.4606	0.0303	0.6728	**0.6876**	49%



(a) Efficiency results of the training phase



(b) Efficiency results of the query phase

Figure 6: Efficiency evaluation of different datasets (in seconds)

the top-k nodes with the highest GNN score or similarity. The hyper-parameters settings are the same as in their original paper.

Query setting. In the experiment, all query nodes are randomly selected to prevent potential bias. The community size k is dependent on users, which can differ among datasets. In OCS, we set 30 as the community size for Facebook, 150 for Cora, Citeseer and Pubmed, and 1000 for MAG and Reddit. In OCIS, as the intersection set is smaller in size, we use $k/5$ as the community size, where k is the community size of the underlying dataset in OCS.

Evaluation metrics. The evaluation of identified communities is conducted through two performance metrics: F1-Score [14, 24], Jaccard similarity (JAC) [30], and overlapping NMI [36]. The F1-Score balances precision and recall, offering a measure of how well the identified community matches the ground truth. JAC evaluates the overlap between the predicted and true communities by comparing the intersection over the union of the two sets. NMI focuses on the alignment between predicted and true overlapping communities, capturing the amount of shared structural information between them. The true data is established as the target community label, with the labels of the identified nodes serving as the predicted data. To evaluate the efficiency, the model training time and online querying time are recorded across different models. All results are averaged across 50 randomly selected queries to ensure the quality of the evaluation process.

Implementation details. SMN is constructed with 16-hop receptive fields, 128 hidden dimensions, and two heads for multi-head attention, using a 64-dimensional SSF. Model training involves a learning rate of 0.02 with 100 and 300 epochs for disjoint and overlapping datasets. Due to its size, SMN is configured with 4-hop receptive fields for MAG and Reddit, while the other hyper-parameters remain the same. The experiments are run on a machine with Intel Xeon 6248R CPU, Nvidia A5000 GPU, and 512GB memory. The code is available at anonymous Github [3].

8.2 Overlapping Community Search

Model effectiveness. Table 2 illustrates the model performance on overlapping community datasets. The comparison is between baseline models, SMN (Sub-Topk), and SMN (Sub-CS) on the Facebook and MAG datasets. Missing results in k-clique and COCLEP are either caused by out-of-memory or failure to assign nodes to

[3]https://anonymous.4open.science/r/SMN-86B4/

Table 3: SMN performance (F1-Score) in Disjoint community search

Datasets	k-clque	CTC	k-core	ICSGNN	QDGNN	COCLEP	SMN
Cora	0.2941	0.3179	0.309	0.7787	0.7208	0.2516	**0.8866**
Citeseer	0.2951	0.281	0.3081	0.7679	0.7486	0.3944	**0.7698**
Pubmed	0.5121	0.5427	0.5586	0.8065	0.7999	0.5153	**0.8255**
Reddit	-	0.1271	0.2681	0.7374	0.8273	-	**0.9433**

any community affiliation. The *Ave+* represents the average performance improvement compared to the proposed SMN against all baseline models. In OCS, instead of selectively choosing a single community as the target, we use each community from the predicted list as the target and report the average performance to avoid bias. In OCIS, we use the full label list of the query as the target set and search for nodes that fall in at least all target communities, where more affiliations are not punished. Among the 9 datasets, Sub-CS demonstrates the best performance on 5 and 6 datasets for OCS and OCIC, respectively. Sub-Topk achieves the best results for others. Notably, the algorithm-based models generate comparatively low performance in F1, JAC, and NMI, respectively. This is mainly caused by the fact that those models are not task-driven and fail to predict the ground truth label.

In the OCS task, the ML-based models exhibit lower performance on the MAG datasets compared to the Facebook datasets. This discrepancy is primarily due to the size and complexity of the datasets. As shown in Table 1, the MAG datasets have relatively lower OR and higher MLA, indicating a higher variance in label distribution. For instance, in the Chemistry dataset, 75% of nodes belong to only one community, while the most popular nodes are associated with up to 13 communities. Despite this significant label variance, the proposed SMN maintained stable and superior performance, achieving *Ave+* improvement of 59%, 58%, and 54% in F1, JAC, and NMI, respectively. These results demonstrate the model's robustness and effectiveness in handling overlapping community structures.

When extending to OCIS, we used the query's full label set as the target community set. In datasets like Chemistry, popular query nodes may be linked to as many as 13 communities, creating a unique challenge that can lead to out-of-sample issues. These issues may hinder the model's ability to identify k nodes that all meet the specific criteria. As shown in Table 1, baseline models experience a significant performance drop when applied to OCIS, whereas SMN demonstrates relatively stable performance across most datasets, with only a slight decline in F1-Score compared to its performance in OCS. This highlights SMN's superiority in handling these more complex scenarios.

F1 and JAC focus on the exact matching of each node's prediction, while NMI emphasizes the similarity between the predicted community and the ground truth community. Despite these differences, the experiment shows similar trends across all three metrics. On average, our proposed SMN model surpasses the best baseline in OCS by 13.50% and in OCIS by 13.96% based on F1-Score, and by 19.19% and 19.95% respectively in terms of NMI. This highlights the effectiveness of SMN in handling diverse community search tasks.

Model training efficiency. Figure 6(a) depicts model training time comparison. The reported training time for SMN includes times for both preprocessing and model training. As ICS-GNN operates in an online setting and is trained and tested on candidate subgraphs, its training time is not included. Notably, QDGNN and COCLEP exhibit significantly longer training time. When training for the

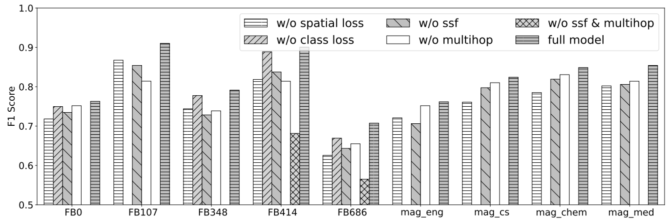


Figure 7: Ablation study

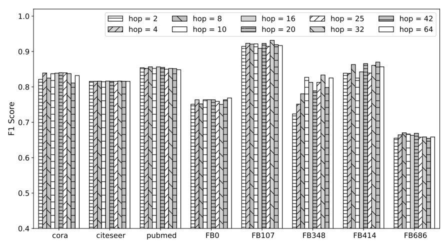
small Facebook datasets by 300 epochs, QDGNN and COCLEP take over 1,000 seconds, whereas SMN requires less than 10 seconds. On the large and densely connected dataset (Reddit), training SMN for 100 epochs takes only 7.6 seconds, compared to QDGNN's 11,055.1 seconds, and COCLEP runs out of memory (OOM). These results empirically demonstrate the efficiency of the proposed lightweight SMN and support Lemma 7.1. The proposed model framework accelerates model training by 2 orders of magnitude on average and achieves up to 3 orders of magnitude on large graphs such as Reddit and MAG-Medicine.

Online query efficiency. Figure 6(b) shows online query performance. QDGNN exhibits slow querying speed due to its reliance on BFS as the backbone algorithm. Overall, Sub-Topk achieves the best results, surpassing the existing best one by 2 orders. Sub-CS demonstrates superior efficiency on smaller datasets compared to COCLEP but slows down on larger datasets due to the necessity of recomputing the centroid each time the community is updated, making it sensitive to community size. The trade-off between Sub-Topk and Sub-CS suggests that Sub-CS is preferable due to its accuracy for complex tasks like OCIS, where the required community size is small. However, for the tasks favoring efficiency with large community sizes, Sub-Topk will be preferred.

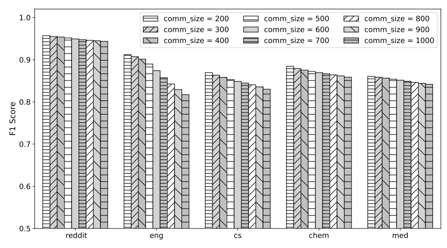
8.3 Disjoint CS and Ablation Study

Disjoint community search. In this section, we report the model's performance on datasets with disjoint communities to further evaluate its effectiveness. The model performance with Sub-CS in disjoint community datasets is presented in Table 3. Although SMN is primarily designed for OCS, it outperforms SOTA models in disjoint community search. SMN consistently achieves superior results compared to all baseline models, showcasing an average improvement of 7.62% across 4 datasets. This enhancement is primarily due to the hop-wise attention mechanism and the proposed search algorithm, which leverage high-order patterns captured from a larger model receptive field. The results prove the effectiveness of SMN in learning representative node embeddings.

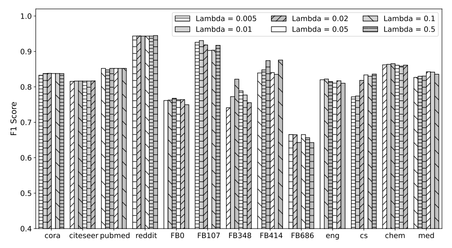
Ablation study. In this section, we investigate the effectiveness of components employed by SMN and SSF in Figure 7 to illustrate the contribution of each design. We conduct the ablation study toward three loss functions, SSF, multi-hop attention, and the full model. Overall, the full model achieves the most stable and superior performance across 9 datasets, while the model without SSF and the multi-hop attention shows the worst performance (mostly below 0.50 F1-Score). We notice that the model without classification loss also performs well in the Facebook datasets. This trend is due to Facebook datasets being smaller in size, which makes performing the classification less challenging. On average, the spatial loss functions and the SSF contribute the most to the model performance, showing improvements of 7.68% and 6.37%.



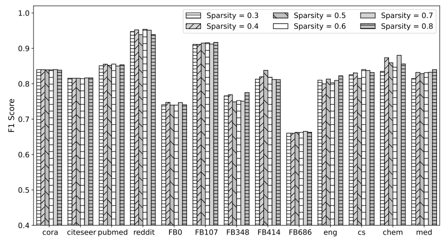
(a) F1-Score by varying hop number



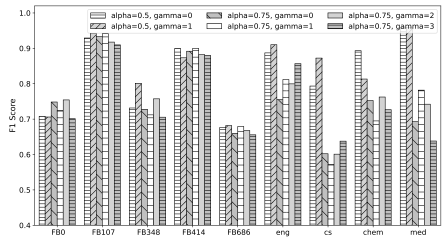
(b) F1-Score by varying community size



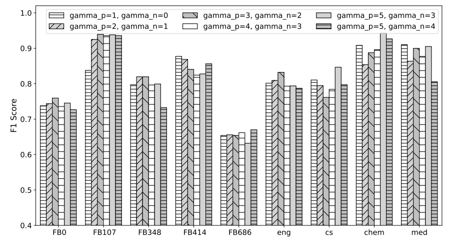
(c) Varying λ in the soft sparse filter



(d) Varying sparsity rate in the hard sparse filter



(e) F1-Score by varying α and γ



(f) F1-Score by varying γ^+ and γ^-

Figure 8: Hyper-parameter analysis

8.4 Hyper-parameter Analysis

In this section, we conduct various experiments on OCS to test the sensitivity of hyper-parameters. The study contains six experiments, covering parameters such as the number of hops, community sizes, λ, α, and γ. Hyper-parameters are tuned using grid search.

Varying hop number. In Figure 8(a), we assess the model's performance by varying hop numbers, which determine the model receptive fields. This experiment evaluates the model's capability of capturing high-order patterns and robustness of oversmoothing. SMN demonstrates a stable and slightly increasing trend as the number of hops increases. Notably, GNN models tend to suffer from oversmoothing, generally limiting the receptive fields to 3 hops. This proves that SMN benefits from higher-order receptive fields by mitigating the oversmoothing effect.

Varying community size. In Figure 8(b), we evaluate the effect of varying community sizes on the model's F1-Score. The community sizes tested range from 200 to 1000 in increments of 100. This experiment evaluates the model's sensitivity to community size, where smaller communities represent a less challenging task compared to larger ones. The results indicate that the model's performance slightly decreases as the community size increases. This demonstrates the model's effectiveness in identifying large communities, as it maintains a high F1-Score even with larger community sizes.

Varying λ in the soft sparse filter. In Figure 8(c), we evaluate the performance of the model by varying the parameter λ in the soft sparse filter. λ is utilized to control the sparsity level in the regulation term in Equation 2. The values of λ tested are 0.005, 0.01, 0.02, 0.05, 0.1, and 0.5. The figure shows that the impact of λ varies across different datasets. The model performance is more sensitive to the λ values for the datasets with overlapping communities.

Varying sparsity rate in the hard sparse filter. Figure 8(d) presents the evaluation results of the model by varying sparsity rates in the hard sparse filter. Unlike using λ to control the sparsity, the hard filter directly zeroes out elements with lower absolute weights in the model classifier based on a predefined sparsity rate. Similar to the findings in above, disjoint datasets do not benefit significantly from sparsity adjustments, whereas the optimal sparsity settings in overlapping datasets are highly dataset-specific.

Varying α and γ. In Figure 8(e), we analyze the effect of varying α and γ on the model's performance. Here, α controls the balance between different loss components, and γ influences the overall weight of the focal losses. The results reported for α are 0.5 and 0.75, and for γ are 0, 1, 2, and 3. This shows that different combinations of α and γ yield varying F1-Score across datasets.

Varying γ^+ and γ^-. Figure 8(f) explores the impact of varying γ^+ and γ^- on the model's F1-Score. These parameters control the influence of positive and negative samples in the training process as shown in Equation 4. The values tested include combinations such as $(\gamma^+ = 1, \gamma^- = 0)$, $(\gamma^+ = 3, \gamma^- = 2)$, and $(\gamma^+ = 5, \gamma^- = 3)$. The results indicate that different settings of γ^+ and γ^- can significantly affect performance. For example, higher values of both γ^+ and γ^- generally lead to better F1-Score in the MAG-Chem and MAG-CS datasets. Conversely, a more balanced setting is preferable in others.

9 CONCLUSION AND FUTURE WORK

This paper studies community search in complex network structures, particularly the challenging domain of overlapping communities (OCS). A general solution of OCS named SSF is proposed, accompanied by a Simplified Multi-hop Attention Network (SMN). The model enables effective exploration of the overlapping community structure within networks. Extensive experiments on 13 real-world datasets prove the superiority of SMN compared to the state-of-the-art approaches across various dimensions, including model effectiveness, training efficiency, and query efficiency. In real-world applications, graphs can be extremely large and evolve over time. One potential limitation of our model is its scalability, as computing high-order adjacency matrices is space-intensive. Future work could focus on optimizing space complexity and improving scalability for large graphs. Additionally, incorporating real-time updates to handle dynamic communities would allow the model to adapt as the network evolves, capturing real-time patterns.

REFERENCES

[1] Esra Akbas and Peixiang Zhao. 2017. Truss-based community search: a truss-equivalence based indexing approach. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1298–1309.

[2] Ali Behrouz, Farnoosh Hashemi, and Laks VS Lakshmanan. 2022. FirmTruss Community Search in Multilayer Networks. *Proceedings of the VLDB Endowment* 16, 3 (2022), 505–518.

[3] Jinyu Cai, Jicong Fan, Wenzhong Guo, Shiping Wang, Yunhe Zhang, and Zhao Zhang. 2022. Efficient deep embedded subspace clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1–10.

[4] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online Search of Overlapping Communities. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, 277–288.

[5] Ehsan Elhamifar and René Vidal. 2013. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence* 35, 11 (2013), 2765–2781.

[6] Shuheng Fang, Kangfei Zhao, Guanghua Li, and Jeffrey Xu Yu. 2023. Community search: a meta-learning approach. In *IEEE 39th ICDE*. 2358–2371.

[7] Yixiang Fang, CK Cheng, Siqiang Luo, and Jiafeng Hu. 2016. Effective community search for large attributed graphs. *Proceedings of the VLDB Endowment* (2016).

[8] Yixiang Fang and Reynold Cheng. 2017. On attributed community search. In *International Workshop on Mobility Analytics for Spatio-temporal and Social Data*. 1–21.

[9] Yixiang Fang, Reynold Cheng, Yankai Chen, Siqiang Luo, and Jiafeng Hu. 2017. Effective and efficient attributed community search. *The VLDB Journal* 26 (2017), 803–828.

[10] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29 (2020), 353–392.

[11] Yixiang Fang, Zheng Wang, Reynold Cheng, Xiaodong Li, Siqiang Luo, Jiafeng Hu, and Xiaojun Chen. 2018. On spatial-aware community search. *IEEE Transactions on Knowledge and Data Engineering* 31, 4 (2018), 783–798.

[12] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3558–3565.

[13] Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. 2016. Top-k overlapping densest subgraphs. *Data Mining and Knowledge Discovery* 30 (2016), 1134–1165.

[14] Jun Gao, Jiazun Chen, Zhao Li, and Ji Zhang. 2021. ICS-GNN: lightweight interactive community search via graph neural network. *Proceedings of the VLDB Endowment* 14, 6 (2021), 1006–1018.

[15] Tao Guo, Xin Cao, and Gao Cong. 2015. Efficient algorithms for answering the m-closest keywords query. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 405–418.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[18] Farnoosh Hashemi, Ali Behrouz, and Milad Rezaei Hajidehi. 2023. CS-TGN: Community Search via Temporal Graph Neural Networks. In *Companion Proceedings of the ACM Web Conference*. 1196–1203.

[19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[20] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying K-Truss Community in Large and Dynamic Graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, 1311–1322.

[21] Xin Huang, Laks VS Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate Closest Community Search in Networks. *Proceedings of the VLDB Endowment* 9, 4 (2015).

[22] Ajay Kumar Jaiswal, Haoyu Ma, Tianlong Chen, Ying Ding, and Zhangyang Wang. 2022. Training your sparse neural network better with any mask. In *International Conference on Machine Learning*. PMLR, 9833–9844.

[23] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. 2017. Deep subspace clustering networks. *Advances in neural information processing systems* 30 (2017).

[24] Yuli Jiang, Yu Rong, Hong Cheng, Xin Huang, Kangfei Zhao, and Junzhou Huang. 2022. Query driven-graph neural networks for community search: from non-attributed, attributed, to interactive attributed. *Proceedings of the VLDB Endowment* 15, 6 (2022), 1243–1255.

[25] Junghoon Kim, Siqiang Luo, Gao Cong, and Wenyuan Yu. 2022. DMCS: Density modularity based community search. In *Proceedings of the 2022 International Conference on Management of Data*. 889–903.

[26] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. (2016).

[27] Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani. 2021. Lassonet: A neural network with feature sparsity. *Journal of Machine Learning Research* 22, 127 (2021), 1–29.

[28] Jure Leskovec and Julian Mcauley. 2012. Learning to discover social circles in ego networks. *Advances in neural information processing systems* 25 (2012).

[29] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9267–9276.

[30] Ling Li, Siqiang Luo, Yuhai Zhao, Caihua Shan, Zhengkui Wang, and Lu Qin. 2023. COCLEP: Contrastive Learning-based Semi-Supervised Community Search. *IEEE 39th ICDE* (2023).

[31] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. 2012. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2012), 171–184.

[32] Siwei Liu, Zaiqiao Meng, Craig Macdonald, and Iadh Ounis. 2023. Graph neural pre-training for recommendation with side information. *ACM Transactions on Information Systems* 41, 3 (2023), 1–28.

[33] Vishal M Patel and René Vidal. 2014. Kernel sparse subspace clustering. In *2014 ieee international conference on image processing (icip)*. 2849–2853.

[34] Tal Ridnik, Emanuel Ben-Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. 2021. Asymmetric loss for multi-label classification. In *Proceedings of the IEEE/CVF international conference on computer vision*. 82–91.

[35] Jingwen Shang, Chaokun Wang, Changping Wang, Gaoyang Guo, and Jun Qian. 2020. An attribute-based community search method with graph refining. *The Journal of Supercomputing* 76 (2020), 7777–7804.

[36] Oleksandr Shchur and Stephan Günnemann. 2019. Overlapping Community Detection with Graph Neural Networks. *Deep Learning on Graphs Workshop, KDD* (2019).

[37] Mauro Sozio and Aristides Gionis. 2010. The Community-Search Problem and How to Plan a Successful Cocktail Party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 939–948.

[38] Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. 2017. Training sparse neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 138–145.

[39] Xingyu Tan, Jingya Qian, Chen Chen, Sima Qing, Yanping Wu, Xiaoyang Wang, and Wenjie Zhang. 2023. Higher-Order Peak Decomposition. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 4310–4314.

[40] Kai-Yu Tang, Ching-Yi Chang, and Gwo-Jen Hwang. 2023. Trends in artificial intelligence-supported e-learning: A systematic review and co-citation network analysis (1998–2019). *Interactive Learning Environments* 31, 4 (2023), 2134–2152.

[41] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. GRAPH ATTENTION NETWORKS. *stat* 1050 (2018), 4.

[42] Hanchen Wang, Rong Hu, Ying Zhang, Lu Qin, Wei Wang, and Wenjie Zhang. 2022. Neural subgraph counting with wasserstein estimator. In *Proceedings of the 2022 International Conference on Management of Data*. 160–175.

[43] H Wang, D Lian, Y Zhang, L Qin, and X Lin. 2021. GoGNN: Graph of graphs neural network for predicting structured entity interactions. (2021).

[44] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Efficient Unsupervised Community Search with Pre-trained Graph Transformer. *arXiv preprint arXiv:2403.18869* (2024).

[45] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Neural Attributed Community Search at Billion Scale. *Proceedings of the ACM on Management of Data* 1, 4 (2024), 1–25.

[46] Yue Wang, Xun Jian, Zhenhua Yang, and Jia Li. 2017. Query optimal k-plex based community in graphs. *Data Science and Engineering* 2 (2017), 257–273.

[47] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. 2021. Dissecting the diffusion process in linear graph convolutional networks. *Advances in Neural Information Processing Systems* 34 (2021), 5758–5769.

[48] Rongzhe Wei, Haoteng Yin, Junteng Jia, Austin R Benson, and Pan Li. 2022. Understanding non-linearity in graph neural networks from the bayesian-inference perspective. *Advances in Neural Information Processing Systems* 35 (2022), 34024–34038.

[49] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. 6861–6871.

[50] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks? (2018).

[51] Le Yang, Haojun Jiang, Ruojin Cai, Yulin Wang, Shiji Song, Gao Huang, and Qi Tian. 2021. Condensenet v2: Sparse feature reactivation for deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

3569–3578.

[52] Jianke Yu, Hanchen Wang, Xiaoyang Wang, Zhao Li, Lu Qin, Wenjie Zhang, Jian Liao, and Ying Zhang. 2023. Group-based fraud detection network on e-commerce platforms. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5463–5475.

[53] Long Yuan, Lu Qin, Wenjie Zhang, Lijun Chang, and Jianye Yang. 2017. Index-based densest clique percolation community search in networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 5 (2017), 922–935.