SLRL: Semi-Supervised Local Community Detection Based on Reinforcement Learning

Li Ni[1,2], Rui Ye[1], Wenjian Luo[3], Yiwen Zhang[1*], Lei Zhang[1], Victor S. Sheng[4]

[1]School of Computer Science and Technology, Anhui University, China
[2]Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, China
[3]School of Computer Science and Technology, Harbin Institute of Technology, China
[4]School of Department of Computer Science, Texas Tech University, USA
nili@ahu.edu.cn, e22301212@stu.ahu.edu.cn, luowenjian@hit.edu.cn, zhangyiwen@ahu.edu.cn, zl@ahu.edu.cn, victor.sheng@ttu.edu

Abstract

Most existing semi-supervised community detection algorithms leverage known communities to learn community structures, subsequently identifying communities that align with these learned community structures. However, differences in community structures may render the community structures learned by these methods inappropriate for the community containing the given node of interest. As a result, the identified community may exclude the given node or be of poor quality. Inspired by the success of reinforcement learning, we propose a Semi-supervised Local community detection method based on Reinforcement Learning, named SLRL, which only explores parts of the network surrounding the given node. It first extracts the local structure around a given node with an extractor, followed by selecting communities that are similar to this local structure to distill useful communities. These selected communities are employed to train the expander, which expands the community containing a given node. Experimental results demonstrate that SLRL outperforms state-of-the-art algorithms on five real-world datasets.

Introduction

Complex systems are often represented as networks (Wang et al. 2023; Chen et al. 2023; Xiong et al. 2024), such as social networks (Girvan and Newman 2002). Real-world networks often exhibit community structures (Fortunato 2010; Zhang et al. 2022; Wu et al. 2021) and some useful prior information exists, such as node label (Liu et al. 2014a,b), pairwise constraints between nodes (Ganji, Bailey, and Stuckey 2018; Li et al. 2014), and real community structures (Zhang et al. 2020), which can enhance the detection of community structures. To utilize prior information, some semi-supervised community detection methods have been devised. For example, Bakshi et al. (Bakshi, Parthasarathy, and Srinivasan 2018) extract the size and patterns of communities from known communities, and subsequently detect communities that are similar to the patterns. Zhang et al. introduced a semi-supervised community detection algorithm based on generative adversarial networks (Zhang et al. 2020). It leverages known communities to identify seed nodes and detects the communities that are similar to the known ones, starting from these identified seed nodes. Wu et al. (Wu et al. 2022) employ a community locator to identify candidate communities similar to known communities and then use a community rewriter to refine these candidates by removing or adding nodes.

The above works (Bakshi, Parthasarathy, and Srinivasan 2018; Zhang et al. 2020; Wu et al. 2022) utilize known communities to detect a specified number of communities from the network that are similar to the known communities. In some cases, users are only interested in the community to which a given node belongs. A simple idea is to directly find the community to which a given node belongs from the multiple communities detected. However, the communities detected by these methods may not contain the given nodes. To mine a community for a given node, Ni et al. (Ni et al. 2024a) proposed a heuristic algorithm that leverages partially known communities to guide the generation of local community to which a given node belongs. Chen et al. (Chen, Xia, and Gao 2023) proposed CommunityAF, which learns a general community structure from known communities and uses an autoregressive flow generative model to generate the community containing a query node. The general community structure learned by CommunityAF may be inappropriate for the community structure containing a given node. So far, the research of semi-supervised local community detection is at the initial stage, necessitating more dedicated efforts urgently.

In this paper, we propose a semi-supervised local community detection algorithm based on reinforcement learning, called SLRL, only exploring parts of the network surrounding the given node. SLRL acquires a local structure around the given node through the extractor. This local structure, which is a rough community, is utilized to select useful communities from known communities. These useful selected communities then form a high-quality training set to train the expander to expand the community. The main contributions are summarized as follows:

- Due to the outstanding performance of reinforcement learning, we employ it for local community detection and propose SLRL. To the best of our knowledge, this is the

*Corresponding author.

first work that applies reinforcement learning to semi-supervised local community detection.

- We design an extractor and expander based on reinforcement learning to extract the local structure and generate the local community, respectively. A strategy based on spectral clustering is designed to select useful communities similar to the extracted local structure from the known communities.
- Experiments are conducted on five real-world datasets. Experimental results show that SLRL achieves better performance than comparison methods.

Related Work

Local Community Detection: Local community detection, which aims to identify the community containing a given node, has garnered much attention (Luo et al. 2018; Ni et al. 2024b, 2023) and is also known as community search (Archer et al. 2017; Huang et al. 2014). For example, Clauset et al. (Clauset 2005) pioneered a local community detection algorithm based on the local modularity R. This method involves adding the node with the maximum increase of modularity to the current community until no such node can be found. Shang et al. (Shang et al. 2023) alternate strong and weak fusion strategies, where strong fusion is used to increase the degree of local modularity of the community, and weak fusion helps the local community to fuse with influential nodes. Roghani et al. (Roghani and Bouyer 2023) calculate the local similarity score of each node and then expand the community starting from the node with a high score. Approaches utilizing user-defined parameters, including k-core (Archer et al. 2017), k-truss (Huang et al. 2014), and k-edge connected component (Zhou et al. 2012), have been employed to detect local communities with specific structures. Some methods utilize random walks to discover local community, such as personalized PageRank-based techniques (Spielman and Teng 2004; Li, Chien, and Milenkovic 2019), local spectral-based method (Li et al. 2015), and heat kernel diffusion approach (Kloster and Gleich 2014). The above studies do not take into account prior information.

Semi-Supervised Community Detection: To take advantage of the prior information, some semi-supervised community detection algorithms have been proposed (Jin et al. 2021). Works (Ganji, Bailey, and Stuckey 2018; Li et al. 2014) take pairwise constraints between nodes as prior information, and works (Liu et al. 2014a,b; Ji et al. 2016) take the labels of nodes as prior information. Besides, recent research has started incorporating known communities as prior information for community detection. Bakshi et al. (Bakshi, Parthasarathy, and Srinivasan 2018) extract several community patterns from known communities. These community patterns are then used to mine communities similar to the known communities. Zhang et al. (Zhang et al. 2020) introduced SEAL, comprising a generator and a discriminator. The generator is designed to generate communities resembling known ones. The discriminator assesses the accuracy of the generated communities, forming a generative adversarial network with the generator. Wu et al. (Wu et al. 2022) presented a framework comprising a community locator and a community rewriter. The locator identifies a specified number of candidate communities similar to the known communities, while the rewriter further refines these candidate communities to achieve higher-quality results. The above works (Bakshi, Parthasarathy, and Srinivasan 2018; Zhang et al. 2020; Wu et al. 2022) are to mine communities similar to known communities from the entire network, which may not encompass the community to which a given node belongs.

Symbol	Definition
G	network
v_0	given node
C	community detected containing v_0
l_{v_0}	local structure around v_0
C_k	known community set
C_{exp}	communities similar to l_{v_0} selected from C_k
$N/N(C_t)$	neighbor node set of community C/C_t
u_t	node selected at the t step
v_{stop}	stop node

Table 1: Important Notations.

Semi-Supervised Local Community Detection: It aims to detect the community to which a given node belongs using known communities. Ni et al. (Ni et al. 2024a) introduced the Semi-supervised Local Community Detection (SLCD) problem. To address the SLCD problem, they proposed SLSS, which leverages the structural similarity between communities to guide the detection of the community containing a given node. Chen et al. proposed a framework called CommunityAF (Chen, Xia, and Gao 2023). CommunityAF leverages Graph Neural Networks to learn node embeddings and employs autoregressive flow generative models to generate the community containing a query node. The latest research indicates that reinforcement learning has demonstrated excellent performance in various domains including healthcare (Yu et al. 2023), finance (Charpentier, Élie, and Remlinger 2020), and computer vision (Le et al. 2022). Inspired by semi-supervised global community detection (Zhang et al. 2020; Wu et al. 2022), this paper employs reinforcement learning for semi-supervised local community detection.

Approach

We first introduce the semi-supervised local community detection problem. For given node v_0, graph $G = (V, E)$, and known community set $C_k = \{c_1, \cdots, c_k\}$. The goal is to detect the community C to which v_0 belongs using known community set C_k (Ni et al. 2024a). Next, we outline the overview of SLRL, followed by a detailed introduction of each step.

Overview

SLRL contains three main steps: extracting local structure, selecting similar communities, and expanding the community. Figure 1 illustrates the process of SLRL. Table 1 presents the main notations utilized in this paper. The input

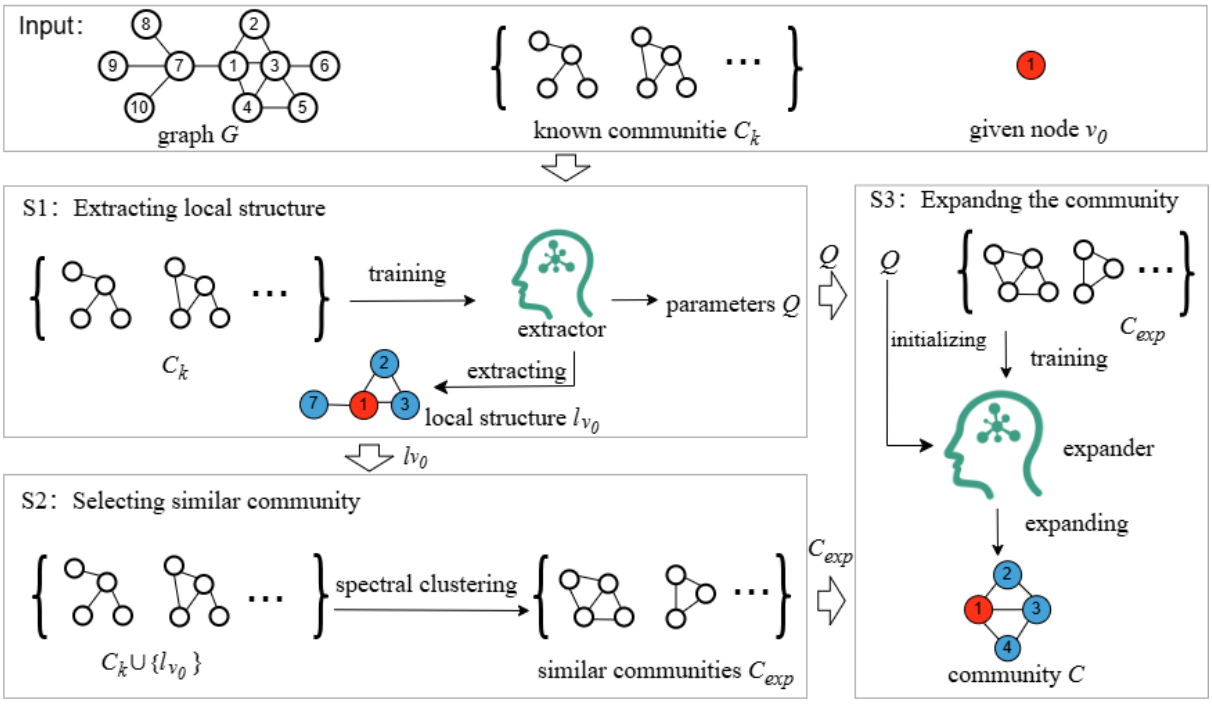


Figure 1: Process of SLRL: extracting local structure, selecting similar communities, and expanding the community.

of SLRL is given node v_0, graph G, and known community set C_k. First, SLRL extracts the local structure l_{v_0} of the given node v_0 using an extractor. Since the extractor trained on the known community set C_k captures the common features of the known communities, the local structure l_{v_0} is actually a rough community containing v_0. There are structural differences within the known communities, and only communities similar to the one containing the given node can guide the detection of the community (Ni et al. 2024a). Next, SLRL uses the extracted structure l_{v_0} to select similar communities C_{exp} from the known communities, distilling useful communities. That is, the local structure l_{v_0} serves as a filter to identify communities similar to the community containing the given node. Finally, these similar communities C_{exp} train an expander to further extract community features in C_{exp} for detecting the community to which a given node belongs.

Extracting Local Structure

We design an extractor to extract the local structure around a given node, which employs reinforcement learning to learn the community structure from known communities. In fact, this local structure is the rough community containing the given node. For better understanding, in this section, we refer to it as the community. The components of the extractor and the optimization process are introduced subsequently.

Design of the Agent The basic idea is that the agent continuously selects nodes from the neighbor nodes of the current community to join the current community, which is inspired by seed expansion methods (Luo, Wang, and Promislow 2008). Adhering to reinforcement learning terminology, we subsequently introduce state, action, and node representation. For the given node v_0, at the t-th step, the current community is $C_{t-1} = \{v_0, u_1, u_2, ..., u_{t-1}\}$, the current state is $S_{t-1} = C_{t-1} \cup N(C_{t-1})$, and the action space is $N(C_{t-1})$, where $N(C_{t-1})$ denotes neighbor nodes of C_{t-1}. The agent picks a node from $N(C_{t-1})$ to join C_{t-1} to obtain community $C_t = \{v_0, u_1, u_2, ..., u_{t-1}, u_t\}$ according to the policy network $\pi(a_t|S_{t-1})$.

To obtain knowledge about the given node and the current community, similar to SEAL (Zhang et al. 2020), at the t-th step, SLRL initializes node representation by indicating whether the node is a given node or belongs to the current community, as follows:

$$\mathbf{z}_t^{(0)}(u) = \mathbb{I}(u = v_0) \cdot \mathbf{q}_1 + \mathbb{I}(u \in C_{t-1}) \cdot \mathbf{q}_2 \quad (1)$$

where $\mathbb{I}$ is an indicator function with value 1 if all its inputs are true and value 0 otherwise. $\mathbf{q}_1$, $\mathbf{q}_2 \in \mathbb{R}^d$ are parameters. Subsequently, the node features are processed by Graph Pointer Network with incremental (iGPN) (Zhang et al. 2020):

$$\mathbf{Z}_t = iGPN(\mathbf{Z}_t^{(0)}) \quad (2)$$

where $\mathbf{Z}_t$ is stacked node representation.

Next, only information about the current community and its neighboring nodes needs to be considered at step t. SLRL uses Multilayer Perceptron (MLP) to obtain the final representation of the relevant nodes:

$$\tilde{\mathbf{Z}}_t(C_{t-1} \cup N(C_{t-1})) = MLP(\mathbf{Z}_t(C_{t-1} \cup N(C_{t-1})); \Psi) \quad (3)$$

where $\tilde{\mathbf{Z}}_t$ represents the final embedding matrix of the nodes, Ψ is a parameter. Additionally, we establish a stop node to determine whether the model should cease expanding the community or not. This stop node aggregates the information on nodes within the current state, as follows:

$$\tilde{\mathbf{z}}_t(stop) = \frac{1}{n} \sum\nolimits_{u \in (C_{t-1} \cup N(C_{t-1}))} \tilde{\mathbf{z}}_t(u) \tag{4}$$

where n is the number of nodes in $C_{t-1} \cup N(C_{t-1})$.

At t step, the next action a_t be executed to select node u_t according to the policy network $\pi(a_t|S_{t-1})$, as follows:

$$\pi(a_t|S_{t-1}) = Softmax(\{\mathbf{q}_3^{\mathrm{T}} \cdot \tilde{\mathbf{z}}_t(u) \mid u \in N(C_{t-1})\}) \tag{5}$$

where $\mathbf{q}_3$ is a parameter.

Optimizing All parameters in the extractor are $\mathbf{Q} = \Psi \cup \{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3\}$. During the optimization process, we assign a reward to each action. The reward received by the agent after performing action a_t is defined as the difference between the $Fscore$ of C_t and the $Fscore$ of C_{t-1} (Wu et al. 2022), expressed as follows:

$$r_t = f(C_t, C_{true}) - f(C_{t-1}, C_{true}) \tag{6}$$

where C_t is the community generated at step t, C_{t-1} is the community generated at step $t-1$, and C_{true} denotes the corresponding ground-truth community.

To maximize the expected cumulative return, we employ policy gradient to update the parameters $\mathbf{Q}$. The specific steps are as follows: 1) Trajectory generating: We choose a batch of nodes from the community in the training set as seed nodes. For each seed node, an extended trajectory $\tau = (S_0, a_1, r_1, ..., S_T)$ can be formed, where S denotes state, a denotes action, and r denotes reward. 2) Gradient computation: For each trajectory, the cumulative reward G_t is computed for each step, as follows:

$$G_t = \sum_{k=0}^{T-t} \gamma^k \cdot r_{t+k} \tag{7}$$

where γ is the discount factor, and T is the end-time step of the trajectory. Subsequently, based on each state-action pair and its corresponding cumulative reward (S, a, G), calculate the policy gradient. 3) Parameter update: Update the parameters $\mathbf{Q}$ based on the computed policy gradients, as follows:

$$\mathbf{Q} = \mathbf{Q} + lr \cdot \sum_{t=1}^{T} \nabla \log(\pi(a_t|S_{t-1})) \cdot G_t \tag{8}$$

Where lr is the learning rate.

The optimization process of the extractor is shown in Algorithm 1. $\mathcal{D}$ contains a set of given nodes and their corresponding true communities from C_k. To prevent the agent from deteriorating and becoming trapped in adversity during certain training batches. After each update of the parameters, using a teacher forcing based on maximum likelihood estimation (MLE) as additional supervised learning (Zhang et al. 2020). This process can efficiently tune the agent.

Algorithm 1: Optimization

Input: G, C_k

```
Initialize agent with parameters Q.
for epochs do
    Generate a training sample set D.
    for Each sample in D do
        Obtain the trajectory τ = (S_0, a_1, r_1, ..., S_T).
        Compute the cumulative reward G_t for each time.
    end for
    Update Q based on Equation (8).
    Teacher-force π using MLE on C_k.
end for
```

After optimization, the extractor is capable of generating the rough community for a given node. The community structure learned by the extractor may not be applicable to the community structure of the given node, so the detected community may be inaccurate. An illustrative example of community expansion is shown in Figure 2. During the training phase, the visited network encompasses known communities and their k-layer boundaries. At step t of the community generation phase, the visited network only includes the current community and its k-layer boundaries, where k corresponds to the number of layers in iGPN. Therefore, the extractor relies solely on the local information of the network throughout the entire community detection process.

Selecting Similar Communities

Since communities with similar structures can enhance the quality of communities, we select communities similar to the local structure l_{v_0}. Among the known communities, communities with similar structures have high structural similarity, while dissimilar communities have low structural similarity. This leads to similar communities displaying cluster characteristics. It is intuitive to use a clustering method to divide the known communities into different clusters and the communities within the same cluster have similar community structures. Therefore, we adopt spectral clustering (Damle, Minden, and Ying 2018) to divide the communities within $C_k \cup \{l_{v_0}\}$ into multiple clusters. Subsequently, communities in the same cluster as l_{v_0} are selected to form C_{exp}, the training set for the expander.

The detailed process of selecting communities is described as follows: SLRL first calculates the similarity between pairwise communities in $C_k \cup \{l_{v_0}\}$ to obtain the similarity matrix. This is because the input of spectral clustering is a similarity matrix. For two communities within $C_k \cup \{l_{v_0}\}$, SLRL obtains their corresponding subgraphs, each comprised of nodes within the community and the edges between them. Then, SLRL utilizes the shortest path kernel (Borgwardt and Kriegel 2005) to compute the similarity between these two subgraphs as the similarity between the two communities. Next, spectral clustering divides $C_k \cup \{l_{v_0}\}$ into multiple clusters, with the number of clusters being determined by the silhouette coefficient (Rousseeuw 1987). Finally, SLRL selects the cluster that encompasses the local structure l_{v_0} and excludes l_{v_0} from this

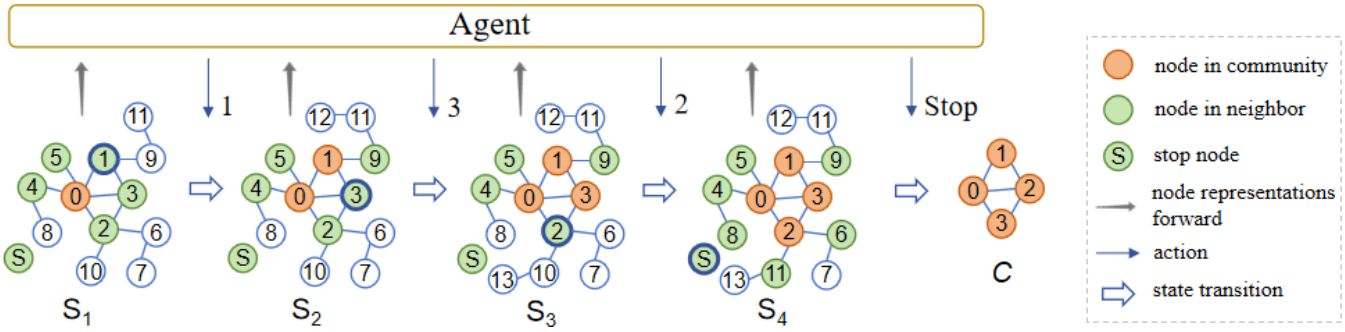


Figure 2: Process of community expansion. Starting with the node v_0, the extractor (expaner) continuously selects a node from the action space and adds it to the community until the stop node is selected or the action space is empty.

Datasets	Nodes	Edges	Number of communities
Amazon	13,178	33,767	4,517
DBLP	114,095	466,761	4,559
LiveJournal	316,606	4,945,140	4,510
Youtube	216,544	1,393,206	2,865
Twitter	87,760	1,293,985	2,838

Table 2: Statistics on real-world datasets.

cluster to obtain similar communities set C_{exp}.

Expanding the Community From v_0

The C_{exp} obtained from the previous section is used to train the expander to expand communities. The design of the expander is the same as that of the extractor. The difference between the expander and the extractor is the use of different training sets. For the optimization process, the extractor is trained on all known communities, while the expander is trained on selected useful communities C_{exp} that are similar to the local structure. Since the extractor has already learned a universal community structure, the initial parameters of the expander are derived from the extractor, which enables the expander to achieve rapid convergence. The community structure learned by the extractor may not be applicable to the community structure of the given node. Therefore, the expander needs to be further tuned by selected known communities C_{exp}. After optimization, the expander is capable of generating communities. The community generation of the expander is the same as that of the extractor, as depicted in Figure 2.

Experiment

In this section, we conduct experiments to evaluate the performance of SLRL, and the code of SLRL is available[1].

Experimental Settings

Dataset: We utilize five real-world datasets, including Amazon (Yang and Leskovec 2012), DBLP (Backstrom et al. 2006), LiveJournal (Backstrom et al. 2006), Youtube (Mislove et al. 2007), and Twitter (McAuley and Leskovec 2012). Table 2 provides a summary of the dataset statistics. In our experiments, the dataset is preprocessed following the work (Zhang et al. 2020). For each dataset, we select given nodes as follows: 1000 nodes from Amazon with an interval of 13 (selecting one node for every 13), 1000 nodes from DBLP with an interval of 114, 500 nodes from LiveJournal with an interval of 633, 500 nodes from Youtube with an interval of 433, and 500 nodes from Twitter with an interval of 41. In addition, the number of clusters is set to 2, which is determined to maximize the silhouette coefficients (Rousseeuw 1987).

Evaluation Metrics: To evaluate the performance of algorithm, we employ various metrics: Precision (Fisher 1936), Recall (Fisher 1936), Fscore (Powers 2011), and Jaccard (Jaccard 1912), to gauge the disparities between the detected communities and the ground-truth communities (Bakshi, Parthasarathy, and Srinivasan 2018; Zhang et al. 2020; Jia et al. 2019). For the semi-supervised global community detection algorithm, the detected community is the community that contains the given node among all the detected communities. The higher metrics indicate the better quality of the detected communities.

Comparison Methods: We compare the proposed method with one local community detection algorithm (M method (Luo, Wang, and Promislow 2008)), two semi-supervised global community detection methods (SEAL (Zhang et al. 2020) and CLARE (Wu et al. 2022)), and two semi-supervised local community detection algorithms (SLSS (Ni et al. 2024a) and CommunityAF (Chen, Xia, and Gao 2023)). The number of communities generated by CLARE and SEAL is set to 5000 on the Amazon, DBLP, LiveJournal, Youtube, and Twitter datasets. Similar to SEAL, SLRL terminates when the community size exceeds the maximum size of known communities. Two groups of ground-truth communities, each comprising 100 communities, are selected as the known communities for the experiments. SEAL, CLARE, CommunityAF, and SLRL are run three times for each group using different random numbers. The average metrics from these three runs are used as the metric for each group. The average metrics across the two groups are taken as the final metrics for the algorithm.

Experimental Results

[1]https://github.com/nilics/AAAI2025-SLRL.git

Datasets	Metrics	M	SLSS	ComAF	SLRL
Amazon	P	0.806	0.846	0.829	**0.854**
	R	0.792	0.868	0.904	**0.935**
	F	0.759	0.828	0.844	**0.878**
	J	0.657	0.737	0.750	**0.798**
DBLP	P	0.599	0.617	0.515	**0.672**
	R	0.596	0.720	**0.801**	0.726
	F	0.554	0.626	0.606	**0.662**
	J	0.441	0.507	0.459	**0.539**
LiveJournal	P	0.569	0.581	**0.665**	0.665
	R	0.726	**0.797**	0.638	0.702
	F	0.609	**0.643**	0.600	0.580
	J	0.516	**0.531**	0.485	0.468
Youtube	P	0.283	0.247	0.352	**0.539**
	R	0.304	**0.370**	0.295	0.272
	F	0.244	0.241	0.277	**0.292**
	J	0.160	0.154	0.180	**0.192**
Twitter	P	0.262	0.317	**0.427**	0.357
	R	**0.747**	0.575	0.343	0.544
	F	0.308	0.377	0.281	**0.378**
	J	0.205	**0.254**	0.179	0.251

Table 3: Results of M, SLSS, CommunityAF, and SLRL. ComAF is the abbreviation of CommunityAF.

Datasets	Metrics	SEAL-SLRL		CLARE-SLRL	
		SEAL	SLRL	CLARE	SLRL
Amazon	P	0.838	**0.863**	0.838	**0.853**
	R	0.883	**0.924**	0.801	**0.934**
	F	0.839	**0.876**	0.795	**0.877**
	J	0.753	**0.798**	0.687	**0.797**
DBLP	P	0.640	**0.685**	0.596	**0.637**
	R	0.675	**0.752**	0.642	**0.741**
	F	0.625	**0.685**	0.596	**0.653**
	J	0.497	**0.563**	0.471	**0.528**
LiveJournal	P	**0.759**	0.681	**0.691**	0.661
	R	0.681	**0.850**	0.658	**0.789**
	F	0.679	**0.682**	0.640	**0.646**
	J	0.553	**0.563**	0.517	**0.527**
Youtube	P	0.446	**0.513**	0.395	**0.479**
	R	0.359	**0.373**	**0.352**	0.313
	F	0.356	**0.361**	**0.322**	0.306
	J	0.249	**0.250**	**0.217**	0.205
Twitter	P	0.330	**0.361**	0.312	**0.388**
	R	0.297	**0.511**	0.326	**0.555**
	F	0.270	**0.330**	0.285	**0.374**
	J	0.166	**0.214**	0.179	**0.249**

Table 4: Results of CLARE, SEAL, and SLRL.

Comparison With Local Community Detection Methods Here, we compare both unsupervised and semi-supervised local community detection methods, including the M method, SLSS, and CommunityAF with SLRL on five real-world datasets. Table 3 presents the results of these methods.

Table 3 shows the performance of SLRL is significantly better than that of M method, SLSS, and CommunityAF. Specifically, on Amazon, DBLP, and Youtube datasets, SLRL outperforms the M method, SLSS, and CommunityAF. SLRL is worse than other methods on the LiveJournal dataset. Except for Youtube (LiveJournal, Twitter) dataset, SLSS (CommunityAF) is better than M method. The reason why SLRL, CommunityAF, and SLSS are superior to the M method on most datasets is that they use known communities to guide community detection. M method shows good performance on the LiveJournal dataset, which indicates that the community structure on the LiveJournal dataset is consistent with the assumptions of the community structure of the M method. Except for the LiveJournal dataset, SLRL is superior to SLSS because it uses reinforcement learning, which automatically learns and captures nonlinear features and patterns in complex network structures. This ability helps to discover community structure more accurately. The poor performance of SLRL on LiveJournal is due to the imbalance in the number of communities with different structures. For example, among the 100 known communities, two distinct types of community structures exist: 17 communities belong to one cluster, while the remaining 83 belong to another. The small sample size of these 17 communities may prevent the algorithm from fully exploring the state space or lead to overfitting, thus affecting the performance of SLRL. SLRL surpasses CommunityAF in performance because it customizes training sets for various given nodes, while CommunityAF relies on a uniform training set for all cases.

Comparison With Semi-Supervised Community Detection Methods As mentioned in the Introduction, CLARE and SEAL may not be able to detect the community to which some given nodes belong. For comparison with CLARE (SEAL), the average Precision, Recall, Fscore, and Jaccard are calculated on nodes that can be detected by CLARE (SEAL). Table 4 shows the results of CLARE, SEAL, and SLRL on five datasets.

Table 4 shows that SLRL outperforms CLARE and SEAL on most datasets. Specifically, SLRL outperforms SEAL on Amazon, DBLP, LiveJournal, Youtube, and Twitter datasets, and outperforms CLARE on Amazon, DBLP, LiveJournal, and Twitter datasets. The reason for the poor performance of CLARE and SEAL may be that CLARE and SEAL learn concepts representing common features from known communities, which may not be suitable for communities containing a given node due to certain differences between community structures. SLRL performs worse than CLARE on the Youtube dataset because CLARE's size restrictions for generated communities more accurately reflect the actual community sizes in this dataset.

Ablation Study

To explore the effectiveness of the steps in the SLRL, we conduct ablation experiments on Amazon and DBLP datasets. Three simplified versions of SLRL are constructed as follows: 1) "w/o. ext". SLRL is simplified by removing

Datasets	Metrics	w/o. ext	w/o. sel	w/o. exp	SLRL
Amazon	Precision	**0.912**	0.883	0.883	0.849
	Recall	0.849	0.868	0.875	**0.934**
	Fscore	0.856	0.852	0.860	**0.873**
	Jaccard	0.770	0.759	0.774	**0.791**
DBLP	Precision	0.545	0.530	0.552	**0.585**
	Recall	0.736	**0.742**	0.716	0.730
	Fscore	0.586	0.577	0.571	**0.613**
	Jaccard	0.460	0.447	0.444	**0.490**

Table 5: Results of ablation study.

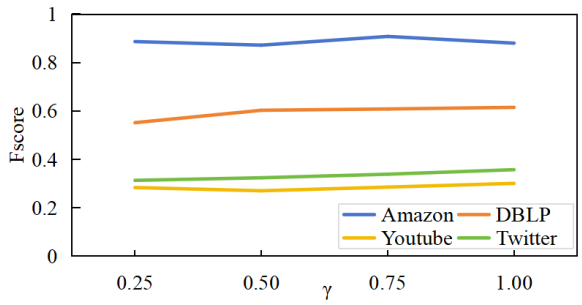


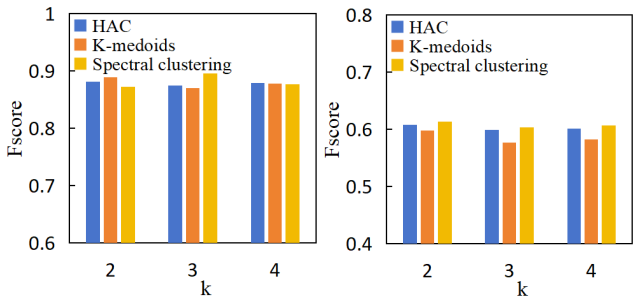
Figure 3: Results of SLRL with different γ.

the step of extracting local structure and using a 1-ego network instead of the local structure. 2) "w/o. sel". SLRL is simplified by removing the step of selecting similar communities and using randomly selected communities instead. 3) "w/o. exp". SLRL is simplified by removing the step of expanding the community. We select 100 nodes as the given nodes to conduct experiments.

Table 5 demonstrates experimental results of w/o. ext, w/o. sel, w/o. exp, and SLRL. Overall, SLRL outperforms w/o. ext, w/o. sel, and w/o. exp, indicating that the steps of extracting local structure, selecting similar communities, and expanding the community play a crucial role. The reason SLRL outperforms w/o. ext is that the similar community structure selected using the 1-ego network may not be suitable for the community containing the given node. SLRL exceeds w/o. sel, and w/o. exp in terms of Fscore and Jaccard, demonstrating the effectiveness of selecting similar communities and expanding the community.

Parameter Study

We explored the effect of the discount factor γ of formula (7) on SLRL performance. Figure 3 presents the Fscore for the Amazon, DBLP, Youtube, and Twitter datasets across different γ values. On the Amazon dataset, the performance of SLRL shows an upward trend as γ increases from 0.25 to 0.75. With a further increase from 0.75 to 1, the performance of SLRL shows a slight decrease. On the DBLP, Youtube, and Twitter datasets, SLRL shows an upward trend as γ increases from 0.25 to 0.75, subsequently stabilizing with a slight increase. During the experiments, we set γ to 1 for all datasets.



(a) Amazon (b) DBLP

Figure 4: Comparison of different clustering algorithms and number of clusters

Effect of Clustering Algorithms and Number of Clusters

We conduct experiments to demonstrate the performance of SLRL under different clustering algorithms and the number of clusters. Specifically, we replaced the spectral clustering (Damle, Minden, and Ying 2018) in SLRL with K-medoids (Park and Jun 2009) and Hierarchical Agglomerative Clustering (HAC) with Complete Linkage (Murtagh 1983) and set the number of clusters to 2, 3, and 4, respectively. 100 nodes are randomly selected on the Amazon and DBLP datasets as given nodes.

Figure 4 shows the results on the Amazon and DBLP datasets. SLRL, when combined with spectral clustering, hierarchical agglomerative clustering, or K-medoids, yields similar Fscore values on the Amazon and DBLP datasets, with differences within 0.02. This suggests that the performance of SLRL is robust to variations in both the clustering algorithm and the number of clusters, demonstrating strong stability across different clustering methods and parameters In the experimental setting, the number of clusters is set to 2, which is determined to maximize the silhouette coefficients (Rousseeuw 1987).

Conclusion

To mine the community containing a given node with some known communities, we propose a method based on reinforcement learning, named SLRL. SLRL first trains an extractor using all known communities to extract a local structure containing a given node. Next, communities similar to the local structure are selected from the known communities as a training set, which is used to optimize the expander for generating a community containing the given node. Extensive experiments on five real-world datasets demonstrate that SLRL outperforms other methods. A limitation of SLRL is that it requires the use of known communities as prior information. In the future, we will continue to explore applying SLRL to mine communities in multiple networks.

Acknowledgements

This work was supported by the National Natural Science Foundation of China [No.62106004, No.62206004,

and No.62272001], Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies [No.2022B1212010005], and the Natural Science Foundation of Anhui Province [No.2408085MF152].

References

Archer, A.; Lattanzi, S.; Likarish, P.; and Vassilvitskii, S. 2017. Indexing Public-Private Graphs. In *Proceedings of the 26th International Conference on World Wide Web*, 1461–1470. Perth, Australia.

Backstrom, L.; Huttenlocher, D. P.; Kleinberg, J. M.; and Lan, X. 2006. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 44–54. Philadelphia, PA, USA.

Bakshi, A.; Parthasarathy, S.; and Srinivasan, K. 2018. Semi-Supervised Community Detection Using Structure and Size. In *2018 IEEE International Conference on Data Mining*, 869–874. Singapore.

Borgwardt, K. M.; and Kriegel, H. 2005. Shortest-Path Kernels on Graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining*, 74–81. Houston, Texas, USA.

Charpentier, A.; Élie, R.; and Remlinger, C. 2020. Reinforcement Learning in Economics and Finance. *Computational Economics*, 62: 425–462.

Chen, J.; Xia, Y.; and Gao, J. 2023. CommunityAF: An Example-based Community Search Method via Autoregressive Flow. *Proceedings of the VLDB Endowment*, 16(10): 2565–2577.

Chen, L.; Wu, L.; Zhang, K.; Hong, R.; Lian, D.; Zhang, Z.; Zhou, J.; and Wang, M. 2023. Improving Recommendation Fairness via Data Augmentation. In *Proceedings of the ACM Web Conference 2023*, 1012–1020. New York, NY, USA.

Clauset, A. 2005. Finding local community structure in networks. *Physical review E*, 72(2): 026132.

Damle, A.; Minden, V.; and Ying, L. 2018. Simple, direct and efficient multi-way spectral clustering. *Information and Inference: A Journal of the IMA*, 8(1): 181–203.

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7: 179–188.

Fortunato, S. 2010. Community detection in graphs. *Physics Reports*, 486(3): 75–174.

Ganji, M.; Bailey, J.; and Stuckey, P. J. 2018. Lagrangian Constrained Community Detection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2983–2990. New Orleans, Louisiana, USA.

Girvan, M.; and Newman, M. E. J. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12): 7821–7826.

Huang, X.; Cheng, H.; Qin, L.; Tian, W.; and Yu, J. X. 2014. Querying k-truss community in large and dynamic graphs. In *International Conference on Management of Data, SIGMOD 2014*, 1311–1322. Snowbird, UT, USA.

Jaccard, P. 1912. The distribution of the flora in the alpine zone. 1. *New Phytologist*, 11(2): 37–50.

Ji, M.; Zhang, D.; Xie, F.; Zhang, Y.; Zhang, Y.; and Yang, J. 2016. Semisupervised Community Detection by Voltage Drops. *Mathematical Problems in Engineering*, 2016: 1–10.

Jia, Y.; Zhang, Q.; Zhang, W.; and Wang, X. 2019. CommunityGAN: Community Detection with Generative Adversarial Nets. In *Proceedings of the International Conference on World Wide Web*, 784–794. San Francisco, CA, USA.

Jin, D.; Wang, X.; He, D.; Dang, J.; and Zhang, W. 2021. Robust Detection of Link Communities With Summary Description in Social Networks. *IEEE Transactions on Knowledge and Data Engineering*, 33(6): 2737–2749.

Kloster, K.; and Gleich, D. F. 2014. Heat kernel based community detection. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1386–1395. NY, USA.

Le, N.; Rathour, V. S.; Yamazaki, K.; Luu, K.; and Savvides, M. 2022. Deep reinforcement learning in computer vision: a comprehensive survey. *Artificial Intelligence Review*, 55(4): 2733–2819.

Li, L.; Du, M.; Liu, G.; Hu, X.; and Wu, G. 2014. Extremal optimization-based semi-supervised algorithm with conflict pairwise constraints for community detection. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 180–187. Beijing, China.

Li, P.; Chien, I. E.; and Milenkovic, O. 2019. Optimizing Generalized PageRank Methods for Seed-Expansion Community Detection. In *Advances in Neural Information Processing Systems*, 11705–11716. Vancouver, BC, Canada.

Li, Y.; He, K.; Bindel, D.; and Hopcroft, J. E. 2015. Uncovering the Small Community Structure in Large Networks: A Local Spectral Approach. In *Proceedings of the 24th International Conference on World Wide Web*, 658–668. Florence, Italy.

Liu, D.; Bai, H.-Y.; Li, H.-J.; and Wang, W.-J. 2014a. Semi-supervised community detection using label propagation. *International Journal of Modern Physics B*, 28(29): 1450208.

Liu, D.; Liu, X.; Wang, W.; and Bai, H. 2014b. Semi-supervised community detection based on discrete potential theory. *Physica A: Statistical Mechanics and its Applications*, 416: 173–182.

Luo, F.; Wang, J. Z.; and Promislow, E. 2008. Exploring local community structures in large networks. *Web Intelligence and Agent Systems: An International Journal*, 6(4): 387–400.

Luo, W.; Zhang, D.; Jiang, H.; Ni, L.; and Hu, Y. 2018. Local Community Detection With the Dynamic Membership Function. *IEEE Transactions on Fuzzy Systems*, 26(5): 3136–3150.

McAuley, J. J.; and Leskovec, J. 2012. Learning to Discover Social Circles in Ego Networks. In *Proceedings of 26th Annual Conference on Neural Information Processing Systems*, 548–556. Lake Tahoe, Nevada, United States.

Mislove, A.; Marcon, M.; Gummadi, P. K.; Druschel, P.; and Bhattacharjee, B. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference*, 29–42. San Diego, California, USA.

Murtagh, F. 1983. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *Comput. J.*, 26(4): 354–359.

Ni, L.; Ge, J.; Zhang, Y.; Luo, W.; and Sheng, V. S. 2024a. Semi-Supervised Local Community Detection. *IEEE Transactions on Knowledge and Data Engineering*, 36(2): 823–839.

Ni, L.; Li, Q.; Zhang, Y.; Luo, W.; and Sheng, V. S. 2024b. LSADEN: Local Spatial-aware Community Detection in Evolving Geo-social Networks. *IEEE Transactions on Knowledge and Data Engineering*, 1–16.

Ni, L.; Xu, H.; Zhang, Y.; and Luo, W. 2023. Spatial-Aware Local Community Detection Guided by Dominance Relation. *IEEE Transactions on Computational Social Systems*, 10(2): 686–699.

Park, H.; and Jun, C. 2009. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2): 3336–3341.

Powers, D. M. W. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *ArXiv*, abs/2010.16061.

Roghani, H.; and Bouyer, A. 2023. A Fast Local Balanced Label Diffusion Algorithm for Community Detection in Social Networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(6): 5472–5484.

Rousseeuw, P. J. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20: 53–65.

Shang, R.; Zhang, W.; Zhang, J.; Jiao, L.; Li, Y.; and Stolkin, R. 2023. Local Community Detection Algorithm Based on Alternating Strategy of Strong Fusion and Weak Fusion. *IEEE Transactions on Cybernetics*, 53(2): 818–831.

Spielman, D. A.; and Teng, S. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, 81–90. Chicago, IL, USA.

Wang, X.; Dong, Y.; Jin, D.; Li, Y.; Wang, L.; and Dang, J. 2023. Augmenting Affective Dependency Graph via Iterative Incongruity Graph Learning for Sarcasm Detection. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, 4702–4710. Washington, DC, USA.

Wu, X.; Xiong, Y.; Zhang, Y.; Jiao, Y.; Shan, C.; Sun, Y.; Zhu, Y.; and Yu, P. S. 2022. CLARE: A Semi-supervised Community Detection Algorithm. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2059–2069. DC, USA.

Wu, Y.; Tardos, J.; Bateni, M.; Linhares, A.; Goncalves de Almeida, F. M.; Montanari, A.; and Norouzi-Fard, A. 2021. Streaming Belief Propagation for Community Detection. In *Advances in Neural Information Processing Systems*, volume 34, 26976–26988.

Xiong, S.; Yang, Y.; Payani, A.; Kerce, J. C.; and Fekri, F. 2024. TEILP: Time Prediction over Knowledge Graphs via Logical Reasoning. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, 16112–16119. Vancouver, Canada.

Yang, J.; and Leskovec, J. 2012. Defining and Evaluating Network Communities Based on Ground-Truth. In *Proceedings of the 12th IEEE International Conference on Data Mining*, 745–754. Brussels, Belgium.

Yu, C.; Liu, J.; Nemati, S.; and Yin, G. 2023. Reinforcement Learning in Healthcare: A Survey. *ACM Computing Surveys*, 55(2): 5:1–5:36.

Zhang, E.; Suter, D.; Truong, G.; and Gilani, S. Z. 2022. Sparse Hypergraph Community Detection Thresholds in Stochastic Block Model. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA.

Zhang, Y.; Xiong, Y.; Ye, Y.; Liu, T.; Wang, W.; Zhu, Y.; and Yu, P. S. 2020. SEAL: Learning Heuristics for Community Detection with Generative Adversarial Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1103–1113. CA, USA.

Zhou, R.; Liu, C.; Yu, J. X.; Liang, W.; Chen, B.; and Li, J. 2012. Finding maximal k-edge-connected subgraphs from a large graph. In *15th International Conference on Extending Database Technology*, 480–491. Berlin, Germany: ACM.