Sattar *et al. Applied Network Science* (2023) 8:64
https://doi.org/10.1007/s41109-023-00592-1

Applied Network Science

RESEARCH Open Access

Exploring temporal community evolution: algorithmic approaches and parallel optimization for dynamic community detection

Naw Safrin Sattar[1*], Aydin Buluc[2], Khaled Z. Ibrahim[2] and Shaikh Arifuzzaman[3*]

*Correspondence:
sattarn@ornl.gov;
shaikh.arifuzzaman@unlv.edu

[1] Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[2] Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[3] University of Nevada, Las Vegas, NV 89154, USA

Abstract

Dynamic (temporal) graphs are a convenient mathematical abstraction for many practical complex systems including social contacts, business transactions, and computer communications. Community discovery is an extensively used graph analysis kernel with rich literature for static graphs. However, community discovery in a dynamic setting is challenging for two specific reasons. Firstly, the notion of temporal community lacks a widely accepted formalization, and only limited work exists on understanding how communities emerge over time. Secondly, the added temporal dimension along with the sheer size of modern graph data necessitates new scalable algorithms. In this paper, we investigate how communities evolve over time based on several graph metrics under a temporal formalization. We compare six different algorithmic approaches for dynamic community detection for their quality and runtime. We identify that a vertex-centric (local) optimization method works as efficiently as the classical modularity-based methods. To its advantage, such local computation allows for the efficient design of parallel algorithms without incurring a significant parallel overhead. Based on this insight, we design a shared-memory parallel algorithm *DyComPar*, which demonstrates between 4 and 18 fold speed-up on a multi-core machine with 20 threads, for several real-world and synthetic graphs from different domains.

Keywords: Dynamic network, Community detection, Community evolution, Temporal network, Permanence, Parallel algorithm, Multi-threading

Introduction

Community detection (Blondel et al. 2008; Lancichinetti and Fortunato 2009; Girvan and Newman 2002) is a fundamental problem in network science with a wide range of applications in different scientific domains. Complex networks are generated at a fast pace nowadays; social, biological, and communication networks are some of the prominent examples. It is difficult to model these networks as static relations because time plays an important role in the evolution of connectivity and patterns. For instance, complex networks in the social domain (e.g., Facebook, Twitter) are fully dynamic, and interaction between the entities changes over time. Such dynamic nature of many real-world

Springer Open

© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

networks makes the problem of community detection even more challenging, as the community structure can change over time.

Community detection is a versatile tool in network science with diverse practical applications (Naik et al. 2022; Karimi et al. 2020). In epidemiology, identifying high-risk communities in epidemic networks can help minimize the spread of diseases such as Covid-19 (Agapito et al. 2022), influenza, HIV, and Ebola. In biological networks, community detection can reveal the functional modules of protein–protein interaction networks (Dilmaghani et al. 2022), gene regulatory networks, and gene co-expression networks. In neuroscience, clustering neural units into densely interconnected groups can help identify functional groups (Martinet et al. 2020).

Understanding the evolution of community structures in complex networks can provide valuable contextual insights. The ability to detect changes in community structures can help detect anomalies or shifts in network behavior. Moreover, predicting future trends of the network can be useful in various domains such as social media (Kazemzadeh et al. 2022), transportation, and public health (Fang et al. 2022). These motivate us to perform the study on the evolving community structures throughout several time intervals in dynamic networks.

Dynamic community detection is challenging and hence an active area of research (Sarıyüce et al. 2016; Halappanavar et al. 2017; Wheatman and Xu 2018; Liu et al. 2020; Li et al. 2020; Pereira et al. 2018; Peixoto and Rosvall 2019; Qiao et al. 2020; Gabert et al. 2021; Ammar 2023; Zou et al. 2023). Many existing works focus on synthetic networks (Zhang et al. 2018; Cazabet et al. 2020; Gabert et al. 2021), which may not fully capture the complexity of real-world networks. Moreover, the rate of edge addition and deletion in real-world networks (Sarıyüce et al. 2016; Pandey et al. 2021; Khanda et al. 2021) may not be uniform, and can be affected by various factors such as seasonality, events, and external interventions. The evaluation and comparison of different dynamic community detection algorithms on the same networks are crucial to understanding their relative strengths and weaknesses. This can help identify the most appropriate method for a given application and network type.

In this study, first, we aim at investigating different dynamic community detection methods. We evaluate six prominent dynamic community detection algorithms. We compare the output communities derived from each of the methods and assess their performance. Based on our findings, we identify a vertex-based metric, to be called *permanence* henceforth, to design a parallel community detection algorithm, *DyComPar*, for dynamic networks. A brief schematic overview of our work is shown in Fig. 1. The use of *permanence* as a vertex-based metric proves useful, as it follows a local optimization approach, unlike approaches based on *modularity*, for instance. This may help avoid the arbitrary tie-breaking scenario that can occur with modularity-based methods. The development of a parallel community detection algorithm, DyComPar, addresses the computational challenges of applying static algorithms repeatedly to evolving networks. Therefore, our study contributes to the understanding of community evolution in complex networks in a scalable manner, with applications including predicting future trends and identifying high-risk communities in disease-spread networks. We anticipate that such a study can be useful for analyzing large dynamic graphs from many emerging data-rich disciplines.

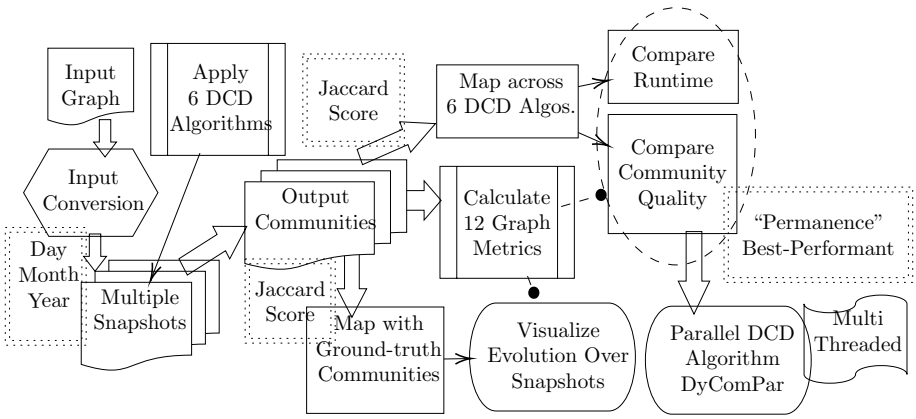

Fig. 1 Workflow for community evolution and parallel community detection in dynamic graphs. We use different real-world and synthetic networks from diverse domains

We summarize the contributions of our work as follows.

- For networks with ground-truth communities, we identify the change of communities in different time phases by computing several graph metrics based on their temporal definition.
- We select the best sequential algorithm by comparing six state-of-the-art dynamic community detection (DCD) methods. Based on the best algorithm, we design our shared memory parallel algorithm for community detection in dynamic graphs using permanence, a vertex-based graph metric for different real-world and synthetic networks from different domains.

The rest of this paper is organized as follows. In "Preliminaries" section we describe in brief the background of dynamic networks and the definition of several graph metrics used for community evolution. We describe the related works in "Related works" section. In "Methodology" section we describe different steps we have performed to understand the evolving community structures per snapshot. We also describe the selection of the DCD algorithms and our method to implement a shared memory parallel algorithm for dynamic community detection using permanence. Our datasets and experimental environment are described in "Experimental setup" section. A detailed analysis of the results is presented in "Results and discussion" section. Finally, "Conclusion" section summarizes the findings and concludes the paper.

Preliminaries

This section provides an explanation of the notations, definitions, and computational models we have used.

Notation

We use dynamic, evolving, and temporal networks interchangeably throughout the paper. We denote the input network as a collection of multiple network snapshots over time for dynamic graphs. There is a change of network in these multiple snapshots for different time frames. We consider multiple edges that appear and

disappear in different time frames. So, the full dynamic network can be represented as $G = G_0 \cup G_1 \cup G_2 \ldots \cup G_t \cup G_{t+1}$

A network snapshot at time t is denoted by $G_t(V_t, E_t)$, where V_t and E_t are the sets of vertices and edges at time t, respectively. Vertices are labeled as $v^0, v^1, \ldots . v^{n-1}$ in different snapshots.. We use the words node and vertex interchangeably as well as links and edges.

Graph metrics for evaluation of dynamic communities

In this section, we describe the graph metrics (Yang and Leskovec 2015) in brief that we use to evaluate the community structures in the dynamic networks. Some metrics only use the community members' internal connectivity, as shown in Table 1. The external connectivity metrics and their definitions are listed in Table 2. Graph metrics that consider both internal and external connectivity are shown in Table 3.

Table 1 Metrics using internal connectivity for dynamic community evolution

Metrics	Function	Definition
Intra-community edges	E_{intra}	Edges between the members of C
Internal density	$\frac{2E_{intra}}{n_c(n_c-1)}$	Internal edge density of the nodes in community C where n_c denotes number of vertices in community C
Average degree	$\frac{2E_{intra}}{n_c}$	Average internal degree of the members of community C
Fraction over median degree (FOMD)	$\frac{\lvert u:u\in C, \lvert(u,v):v\in C\rvert > d_m\rvert}{n_c}$	Fraction of nodes of C that have an internal degree higher than d_m where d_m is the median value of the full graph

Table 2 Metrics using external connectivity for dynamic community evolution

Metrics	Function	Definition
Inter-community edges	E_{inter}	Number of edges on the boundary of C
Expansion	$\frac{E_{inter}}{n_c}$	Number of edges per node that point outside the community
Cut ratio	$\frac{E_{inter}}{n_c(n-n_c)}$	Fraction of existing edges (out of all possible edges) leaving the community, where n is total number of vertices

Table 3 Metrics using both internal and external connectivity for dynamic community evolution

Metrics	Function	Definition
Normalized cut	$\frac{E_{inter}}{2E_{intra}+E_{inter}} + \frac{E_{inter}}{2(E-E_{intra})+E_{inter}}$	Computes the cut cost as a fraction of the total edge connections to all the nodes in the graph
Conductance	$\frac{E_{inter}}{2E_{intra}+E_{inter}}$	Fraction of total edge volume that points outside the community
Clustering coefficient	$CC(v) = \frac{e_i}{\binom{D(v)}{2}}$	Measures the "clumpiness" of a graph, the ratio of the existing edges and the total number of possible edges among the neighbors of a node. Where, e_i : number of edges between the neighbors, and $\binom{D(v)}{2}$: the total number of possible connections among the neighbors
Separability	$\frac{E_{intra}}{E_{inter}}$	Measures the ratio between the internal and the external number of edges on the boundary of C

Table 4 Symbols used for calculating permanence in Eq. 1

Symbol	Meaning
$Perm(v)$	Permanence of vertex v
$I(v)$	Internal neighbors of vertex v
$E_{max}(v)$	Maximum connections to a single external community
C_{in}	Internal clustering coefficient
$C_{in} = \frac{x}{C(I(v),2)}$	x = existing connections among the internal neighbors of v and $C(I(v), 2)$ means the total number of possible connections among the internal neighbors of v
$Perm(v) = 1$	v strongly connected to assigned community
$Perm(v) = 0$	v equally pulled by all neighbors (singleton community)
$Perm(v) = -1$	v weakly connected to assigned community (wrong community)

Permanence

Permanence is a local vertex-based metric used to extract the community structure of large networks based on permanence optimization. It is better in the sense that it can be computed locally and does not require global optimization such as other modularity-based methods. It outperforms well-known community detection methods in terms of runtime complexity as shown in Agarwal et al. (2018). Permanence, *Perm* is calculated using Eq. 1, where $-1 \leq Perm \leq 1$. The meanings are described in Table 4.

$$Perm(v) = \left[\frac{I(v)}{E_{max}(v)} \times \frac{1}{d(v)}\right] - [1 - C_{in}] \tag{1}$$

Related works

Community detection has been a well-known problem in graph mining for a long time (Newman and Girvan 2004; Guo et al. 2014; Sattar and Arifuzzaman 2018a, b, 2020; Sattar 2019, 2022, 2021; Mucha et al. 2010; Sattar and Arifuzzaman 2022). While there are few works on dynamic community detection using various methods (Pereira et al. 2018; Qiao et al. 2020) in the literature, many studies focus on designing efficient data structures for processing dynamic graphs (Ammar 2023; Zou et al. 2023; Green and Bader 2016; Feng et al. 2015). In contrast to the community detection problem, a different study on temporal graph (Bautista and Latapy 2023) focuses on developing a framework for analyzing link streams. This framework can be beneficial for understanding the dynamics and structures of temporal graphs.

However, there are very few works similar to ours performed in this study. Previous studies have focused on progressively evolving graphs only (Cazabet et al. 2020) or evaluated community quality based on a single graph metric conductance (Badlani et al. 2018). Authors in Cazabet et al. (2020) experiment on very small synthetic graphs with a higher number of snapshots. On average those synthetic networks have only 100 vertices and 1200 steps (snapshots). They have shown an evaluation of the smoothness of dynamic partitions (snapshots). They have

compared the output community based on modularity (Newman and Girvan 2004), Adjusted Mutual Information (AMI), and Adjusted Rand Index (ARI). For AMI and ARI calculations what they consider as ground truth is not mentioned in the work. In our work, we include both real-world as well as synthetic networks from different domains. In our analysis, we select networks from varied size ranges, starting with a few hundred vertices and increasing up to millions of vertices and edges. We compare the community quality depending on 12 different graph metrics. We also include the runtime comparison of the 6 DCD methods and choose the best-performing algorithm to present a scalable parallel algorithm for dynamic community detection, *DyComPar*. In Badlani et al. (2018) authors show community quality evaluation at each time slice based on a single graph metric conductance. The value of conductance is compared for each community across the time slices to measure how community structure changes over time. Other than this, no prior works analyzed the community structure quality over the snapshots. In our case, we include 12 different graph metrics to measure how community evolution is taking place per snapshot.

In another study, similar to ours, the authors have compared different static CD methods only for temporal social networks by applying the static methods in multiple snapshots of a network (Rajita et al. 2021). This is similar to the "No Smoothing" technique, where including the Louvain method, they apply some other static CD methods. Their experiments involve using a single dataset only, whereas we compare different dynamic community detection methods for networks from multiple domains.

In a prior study Chakraborty et al. (2017), the authors conducted a comprehensive assessment involving a range of metrics including modularity, conductance, permanence, and others, to determine the most effective metric for static community detection. Within this context, their findings emphasized the potential of permanence as a robust optimization function, showcasing its superiority over alternative metrics in the context of static community detection. Building upon this foundational work, our investigation diverges by encompassing a broader spectrum of metrics and dynamic network scenarios. Through rigorous experimentation across diverse dynamic networks, we arrive at a notable observation: the applicability of permanence as a discriminative metric extends beyond static scenarios. Indeed, permanence emerges as a potent measure, aptly suited for identifying and characterizing dynamic communities. In essence, our study highlights the enduring significance of permanence across dynamic contexts, thereby contributing to the broader understanding of its utility as a pivotal discriminator in the dynamic community detection domain. Our study involves evaluating various metrics and experimenting with different dynamic networks. Our observations indicate that permanence is a reliable measure for detecting dynamic communities.

Overall, our study is unique as we analyze community quality per snapshot based on 12 different graph metrics. Our analysis includes real-world as well as synthetic

networks from different domains, varying in size from a few hundred vertices to millions of vertices and edges. We compare the output community structure of six different dynamic community detection methods and choose the best-performing algorithm to present a scalable parallel algorithm for dynamic community detection, *DyComPar*.

Methodology

In this section, we will outline the various steps we took to observe how ground truth communities evolved in specific networks. Additionally, we will detail the different methods we used and select the optimal DCD algorithm to design our parallel DCD algorithm, *DyComPar*.

Community evolution for ground truth communities

Pre-processing input graph

In our dynamic network experiments, we use the edge-list network format. The input data includes the tuple (source-node, destination-node, timestamp). Depending on the timestamp value and the network duration, we obtain various numbers of snapshots from the input graphs. Networks with connections spanning several years are split into yearly snapshots, while shorter timeframes are divided into monthly snapshots. If the input network has a specified number of ground truth communities per snapshot, we do not need to divide it into multiple snapshots in the pre-processing step. This step is unnecessary for synthetic networks as well, as we generate them using the input parameter for the number of snapshots.

Mapping ground truth communities per snapshot

In order to identify the evolution of ground truth communities throughout the snapshots, we map the communities with one another in all of the snapshots based on Jaccard scorethat is widely accepted and used in several studies for community mapping (Duan et al. 2009; Badlani et al. 2018). We consider choosing 10 communities from the last snapshot and backtracing them in the previous snapshots. We select the 10 communities in the last snapshot based on the largest number of members in each community. We choose communities from the latest snapshot since they tend to be more stable. Conversely, communities in the initial snapshot are more prone to deformation because of the network changes. This observance is shown with an example in " Understanding the evolution of ground truth communities based on different graph metrics" section. Mapping ensures that the evolution of the same community is portrayed throughout the snapshots.

Evaluating community structures using graph metrics per snapshot

Next, we compute the graph metrics values described in "Graph metrics for evaluation of dynamic communities" section for each of the communities. Based on these values we evaluate the community structures per snapshot to understand the evolution of the ground truth communities.

Best sequential algorithm selection from different DCD algorithms

Several algorithms have been proposed in the scientific literature for the detection of dynamic communities within complex networks. In order to ensure a comprehensive analysis, our selection of algorithms is predicated on the prevalent and widely adopted modularity metric. However, recognizing the need for a holistic evaluation, we also incorporate alternative algorithms that leverage distinct metrics. This approach enables us to conduct a rigorous empirical comparison of the various methodologies. In the course of our investigation, we employ six distinct Dynamic Community Detection (DCD) algorithms to meticulously assess and contrast the community quality engendered by each algorithm. This assessment is grounded in a suite of graph metrics elaborated upon in "Graph metrics for evaluation of dynamic communities" section. Concurrently, we extend our inquiry to encompass the runtime performance of each algorithm. The suite of DCD algorithms embraced in this study is succinctly expounded upon below.

No Smoothing Louvain (NoSL) The No Smoothing Louvain (NoSL) method (Cazabet et al. 2020) is a community detection method for evolving networks, where the static Louvain (Newman and Girvan 2004) community detection algorithm is applied at each snapshot of the network without any smoothing or information propagation from the previous snapshots. The Louvain algorithm is a well-known and widely used community detection algorithm that optimizes the modularity of the network to detect communities. In NoSL, the community structure obtained from the Louvain algorithm at each snapshot is considered the community structure of that snapshot without any further modifications. This means that the community structure of each snapshot is independent of the community structure of the previous snapshots, which may not be ideal for evolving networks with gradual changes in community structure over time. However, this approach is computationally efficient and easy to implement, making it useful for such evolving networks where computational resources are limited.

Smoothed Louvain (SmoL) SmoL is a modification of the NoSL method where instead of starting from scratch in every snapshot, the community structure from the previous snapshot is used as the initial guess for the community structure of the current snapshot. The method then optimizes the modularity locally in the same way as the Louvain method. This approach ensures that the community structure evolves smoothly over time and reduces the noise in the community detection results.

DynaMo DynaMo is an adaptive and incremental algorithm that updates the community structure of evolving networks in a non-overlapping manner (Zhuang et al. 2019). Its two-step approach initializes an intermediate community structure based on incremental network changes and the previous network community structure, followed by local modularity optimization and network compression until no further modularity gain improvement is possible.

Algorithm 1: DyComPar: Our Shared-memory-based Parallel Dynamic Community Detection Algorithm

Data: Initial Graph at t $G_t(V_t, E_t)$, Graph at (t+1) $G_{t+1}(V_{t+1}, E_{t+1})$, Initial Community C_t
Result: Community at next time-stamp C_{t+1}

```
edges _added ← E_{t+1} − E_t
edges _deleted ← E_t − E_{t+1}
comm_list ← C_t
for e in edges _deleted do
    /* Intra-Community */
    if C(u) = C(v) then
        if degree(u) = 1&degree(v) = 1 then
            G_t.remove_edge(u, v)
            /* parallel for loop: look up */
            C(u) ← C(u) − u − v
            comm_list ← comm_list ∪ u ∪ v
        else if degree(u|v) = 1 then
            G_t.remove_edge(u, v)
            /* parallel for loop: look up */
            C(u)|C(v) ← C(u) − u|C(v) − v
            comm_list ← comm_list ∪ u|v
        else
            /* Compute-intensive parallel operations; multiple
               parallel loops */
            (perm, comm_list) ← check_permanence()
    /* Inter-Community */
    else
        G_t.remove_edge(u, v)
for e in edges _added do
    /* Intra-Community */
    if C(u) = C(v) then
        G_t.add_edge(u, v)
    /* Inter-Community */
    else
        /* Compute-intensive parallel operations; multiple parallel
           loops */
        (perm, comm_list) ← check_permanence()
C_{t+1} ← comm_list
return C_{t+1}
```

Estrangement Confinement(EsCon) In the Estrangement Confinement (Kawadia and Sreenivasan 2012) method, the idea of the estrangement metric is used to detect communities in dynamic networks. The estrangement metric is a measure of partition distance based on the inertia of inter-node relationships. It quantifies the extent to which two nodes are interacting with each other within a community. In this method, the goal is to maximize the estrangement between nodes in different communities and minimize the estrangement within a community. In the dynamic setting, the algorithm works by initially dividing the nodes into singleton communities and iteratively merging communities based on the estrangement metric. The algorithm takes into account both the current and previous snapshots of the network to update the communities. This method uses a constrained optimization approach to ensure that the resulting communities are meaningful and have a high estrangement value.

Transversal-Network (TraN) In this method, (Mucha et al. 2010) communities at snapshot t depend on the previous $t-1$ and later $t+1$ snapshots of the network. So, a single transversal network is formed by adding inter-snapshot coupling links between the previous $t-1$ and later $t+1$ snapshots of the network. On this network, an adapted version of the modularity optimization algorithm is applied.

DyPerm The DyPerm algorithm (Agarwal et al. 2018) relies on the initial information of the network and community structure to detect communities in the subsequent snapshots. It takes into account the addition and removal of vertices or edges in the subsequent snapshots to update the community structure.

Designing parallel *DyComPar* algorithm

We have described the design of our shared-memory-based parallel algorithm, DyComPar in Algorithm 1. we parallelize the computationally intensive tasks with loop parallelization, and atomic updates.

When we need to iterate over the entire network or the neighbors of a vertex, the members in a community, or any other list, we distribute the work among multiple threads to reduce the workload and perform computations faster, given the large network size. From Algorithm 1, we get the simplistic idea where the parallelization is applied. We have not included a detailed description of each of the computations, rather provide an overview in brief. DyComPar takes as input the initial snapshot of the graph, its subsequent snapshot, and the community information of the initial snapshot. Based on this information, the community assignment for the subsequent snapshot is calculated. If the network has ground-truth community assignments, it is provided as input. For networks with no known ground-truth, the community assignments from the static Louvain method are provided as the initial community assignment. The dynamic changes are captured by identifying the edges being added or deleted in the current snapshot. We maintain two separate lists to separate the deletions and additions of edges. The deleted edges are processed first, as given in line 4, to ensure the graph size is kept smaller with the deletions applied, making the vertices list or community list smaller for the scenarios where a vertex is deleted. Then each edge from the addition list is processed. During edge deletions, if the edge is an inter-community edge, computation of permanence is not needed, and only updating the network occurs. For an intra-community edge, there are three different cases, and only if the vertices of the edges have a non-unit degree, permanence maximization is required to decide the community. For the unit-degree cases, to append or delete the communities from the list, there is a need to iterate over the list where we apply parallelization given in lines 9, and 13 of Algorithm 1. The permanence maximization for community assignments is given in Algorithm 2. Here the source and destination vertices, u and v respectively check for the maximum gain in the permanence value by iterating the neighbors, and checking the permanence value in the neighbor's community, which is parallelized.

Algorithm 2: check_permanence()

```
/* parallel loop */
for k in u|v's neighbor_list do
    check maximum gain in permanence value and move to that community
    community_list ← Update list
```

Algorithm 3: permanence_comm(c)

```
member ← list(c)
/* parallel loop reduction: sum */
for i in member_size do
    sum+ = permanence_node(member[i])
Perm(c) = sum/member.size()
return Perm(c)
```

For checking the gain in permanence value, the permanence value of the community needs to be calculated. This is done by computing the permanence value of each member in the community, and by parallel reduction operation done in Algorithm 3. The computation for the permanence value of a vertex is shown in Algorithm 4. The counting of the number of connections among the internal neighbors of the vertex is done parallelly with atomic update. For edge addition, in the case of intra-community edge, no parallel computation is needed. In the case of inter-community edge, permanence computation and community assignment update are done similarly as given by Algorithm 2.

Algorithm 4: permanence_node(k)

```
/* parallel loop */
for i in comm_size do
    x ←count_connection_internal_neighbor/* atomic update */
den = num_internal_neigh * (num_internal_neigh − 1)/2
Perm(k) = [I(k)/E_max(k) × 1/d(k)] − [1 − x/den]
return Perm(k)
```

Experimental setup

In this section, we provide details on the experimental environments and datasets used in our experiments.

Environment

The experiments for community evolution are run on a Desktop computer with the following specifications: Intel Core i7-4770 CPU 3.40 GHz × 8 Processor, 16 GB of RAM, 1 TB hard disk, and Ubuntu 16.04 LTS.

To parallelize the tasks in a multi-threaded environment for our shared memory implementation, we use the Python Joblib and Multiprocessing modules. The experiments are

performed on the Louisiana Optical Network Infrastructure (LONI), specifically on QB2 (QB2 http://www.hpc.lsu.edu/docs/guides.php? system=QB2), which is a cluster with a peak performance of 1.5 Petaflops and contains 504 compute nodes with over 10,000 Intel Xeon processing cores of 2.8 GHz. Due to the 20 cores per node limitation on QB2, we have used a maximum number of 20 threads. Our approach is similar to the C++ OpenMP implementation.

Dataset

We use both real-world and synthetic networks for our experiments. The real-world networks are depicted in Table 5. For synthetic networks we use the dynamic LFR

Table 5 Real-world networks used in the experiment

Network	Vertices	Edges	Snapshots	Time interval	Description
Collaboration networks					
Cumulative co-authorship	708,497	1,166,376	17	Cumulative years	Cumulatively Aggregated (year) undirected co-authorship network in DBLP repository from 1960 to 2009 (vertex: author, edge: a pair of authors are co-authors at least once) (Chakrabort et al. 2013)
Non-cumulative co-authorship	708,497	1,166,376	17	1 year	Undirected co-authorship network in DBLP repository from 1960 to 2009 (vertex: author, edge: a pair of authors are co-authors at least once) (Chakraborty et al. 2014)
Social networks					
CollegeMsg	1899	59,835	7	193 days (month)	Messages on a Facebook-like platform at UC-Irvine (vertex: user, edge: private message between users at t timestamp) (Leskovec and Krevl 2014)
fb-forum	899	33,720	7	193 days (month)	This network focuses on users' activity in the forum rather than private messages exchanged among users (vertex: user, edge: users' activity in the forum) (Rossi and Ahmed 2015)
Primary school	242	77,602	6	20 s	Contacts between the children and teachers (DATASETS http://www.sociopatterns.org/datasets; Stehlé et al. 2011; Gemmetto et al. 2014)
Citation network					
cit-HepTh	22,768	352,807	7	10 years (year)	Arxiv HEP-TH (high energy physics theory) citation graph is from arXiv and covers all the citations from April 1993 to 2003. (vertex: paper, edge: a pair of papers) Edges from u to v indicate that the paper u cited another paper v (Rossi and Ahmed 2015)

Table 6 Synthetic networks generated using LFR benchmarks used in the experiment

Network	*N*	μ	*s*
Syn-1	3500	0.2	20
Syn-2	1000	0.2	30
Syn-3	1000	0.2	10

benchmark model (Lancichinetti et al. 2008) with parameters Number of Vertices (N), Mixing Coefficient (μ) and Number of Snapshots (s). The parameters for the synthetic networks are given in Table 6.

Results and discussion

In this section, we present the findings of our experiments.

Understanding the evolution of ground truth communities based on different graph metrics

Community mapping in each of the snapshots is crucial to understand the community evolution over snapshots. We observe that communities in the last snapshot are more stable compared to the first snapshot. So we have backtraced the stable communities from the last snapshot to the first snapshots to track their evolution. We have shown this for the ground truth communities for the Primary School network and a synthetic network given in Fig. 2.

If we map communities with respect to the first snapshot, all communities have a very low value of the Jaccard coefficient in the following snapshots observed in Fig. 2a. Instead, if we map communities with respect to the last snapshot, the communities show better values for the Jaccard coefficient in the previous snapshots. There is a mix-up of high and low Jaccard scores found in Fig. 2b. While we experiment with this similar mapping with the synthetic networks, we observe that mapping w.r.t. the first (Fig. 2c) or last (Fig. 2d) snapshot does not show any significant difference drop in the Jaccard values. As in our study, we emphasize real-world networks, that's why we map communities in the reverse order. This makes it easier to follow the evolution in communities.

Community Size In the Primary School network (Fig. 3a), the selected communities are mostly constant in size per snapshot. Only communities with id 4, and (8, 9) shrink in size in time-slice 2 and 6 respectively. In the Cumulative CoAuthorship network (Fig. 3b), the selected communities increase in size (mostly linearly, exponentially) per snapshot. In the Syn-1 network (Fig. 3c), the selected communities follow a mixed trend. The increase or decrease is very minimal for most communities. In the Syn-3 network (Fig. 3d), the selected communities also follow a mixed trend similar to the Syn-1 network. The increase or decrease varies within the range of 2 to 9 for each snapshot.

Intra-community edge

In the Primary School network (Fig. 4a), the values of the intra-community edge of the community members for the selected communities follow a mixed trend. All of the communities have a decrease in snapshot 2. A few decreases can be observed in snapshot 4 (community id 5, 9). In snapshot 6, 50% of the communities have a decreased value and

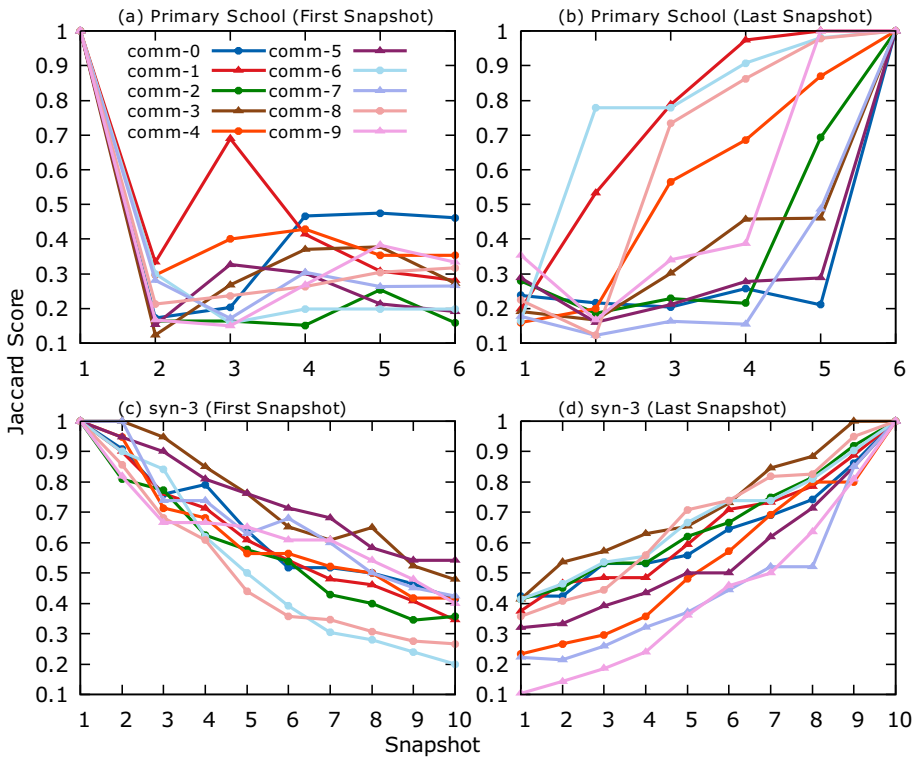

Fig. 2 Community mapping using Jaccard coefficient for the ground truth communities of Primary School network (**a**, **b**) and a synthetic network Syn-3 (**c**, **d**)

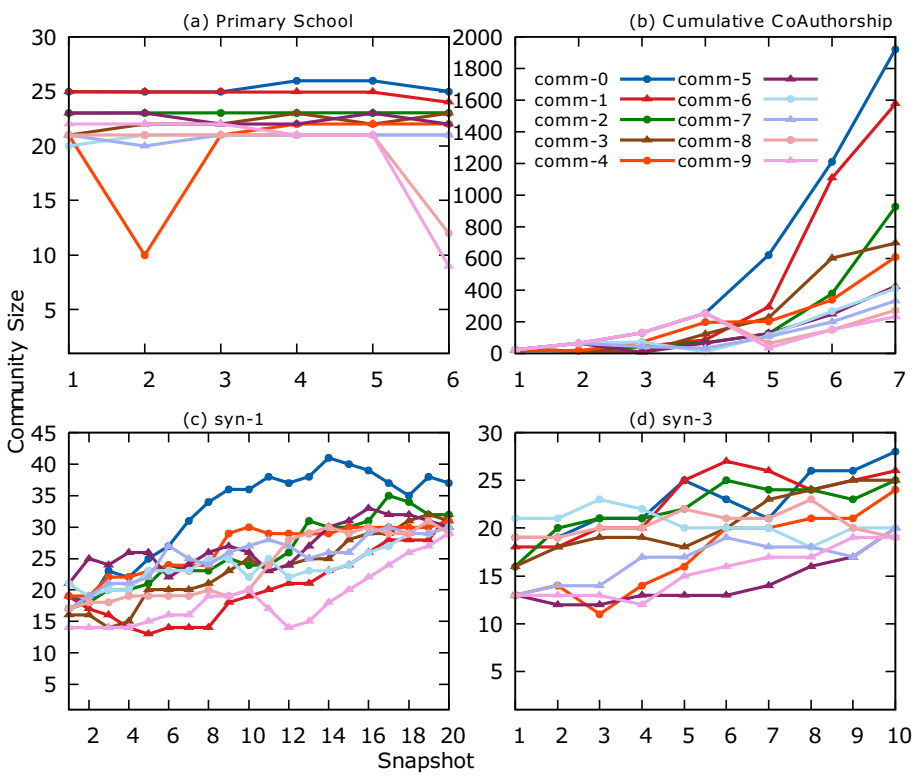

Fig. 3 Size of top 10 communities for different networks (a-d) in distinct snapshots of the networks

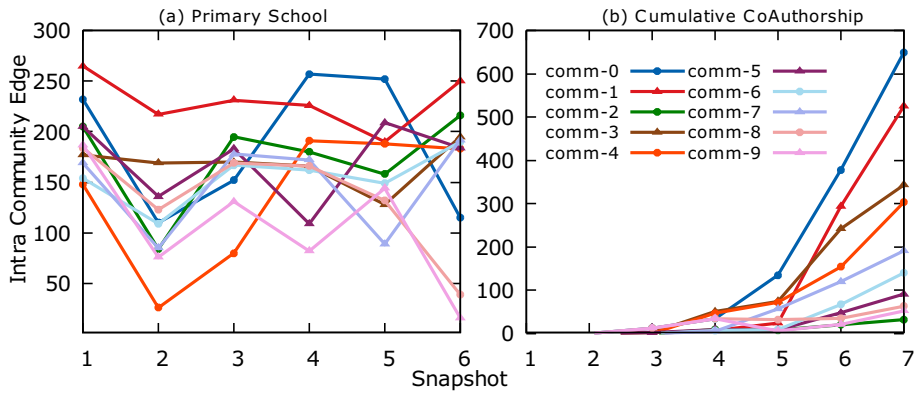

Fig. 4 Intra-community edges for different networks (**a**, **b**) in distinct snapshots of the networks. Syn-1 and Syn-3 networks show constant insignificant values (0/1) over all snapshots

the rest remain constant (30%) or increase (20%). In the Cumulative Co-Authorship network (Fig. 4b), the intra-community edges for the selected communities increase mostly linearly, power (community id 0), and logarithmic (community id 1). The increase is more significant in the last two snapshots.

Internal density

In the Primary School network (Fig. 5a), the values of the internal edge density of the community members for the selected communities mostly follow a zig-zagged pattern.

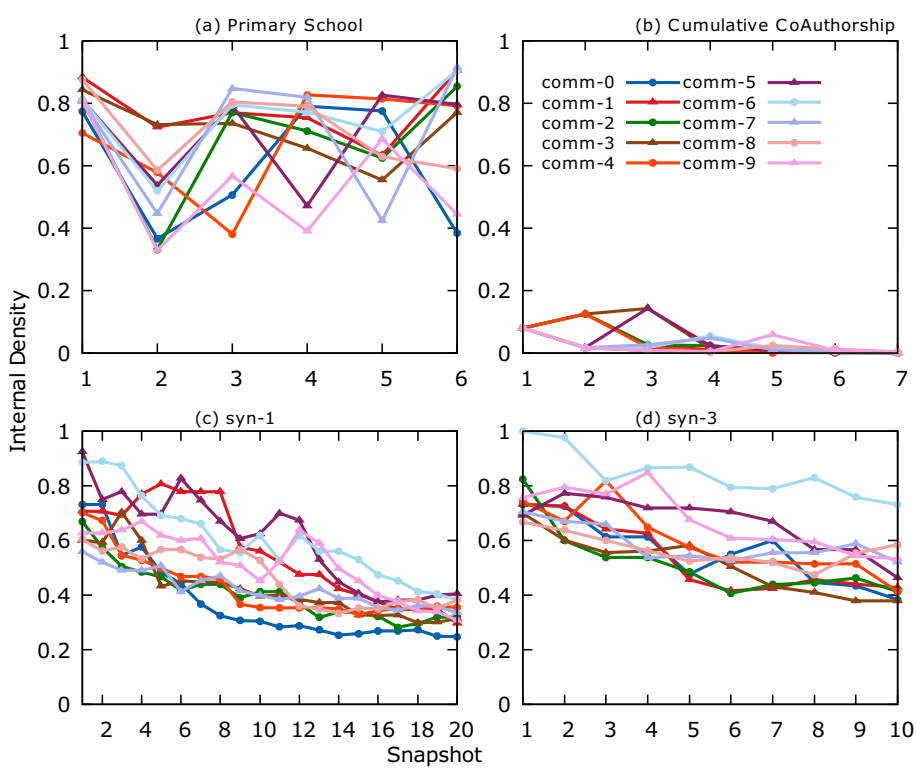

Fig. 5 Internal density values for different networks (**a**–**d**) in distinct snapshots of the networks

Half of the communities have values above average (0.5 to 1) indicating a good quality of the communities throughout the snapshots. In the Cumulative CoAuthorship network (Fig. 5b), the values are within the range of 0.01–0.15 and can be considered negligible. For this network, the size of the communities grows, so does the number of intra-community edges. The range of increase is also high compared to the other networks. This results in a small value for the internal density ratio, such as 0.0006 for community 4 (calculated as 2 multiplied by 300, divided by 1000, and then by 999). As a result, this metric does not provide much insight into the strengths or weakness of the connections among community members. In the Syn-1 network (Fig. 5c), the selected communities follow mostly a decreasing trend of internal edge density of the community members over time, with a few spikes in some of the snapshots. In the Syn-3 network (Fig. 5d), the selected communities follow a similar pattern to the Syn-1 network. Only communities with ids 0,6, and 9 show a spike in snapshots 7,4, and 8 respectively.

Average degree

In the Primary School network (Fig. 6a), the average internal degrees for the selected communities mostly follow a similar pattern as the intra-community edge metric. The average degree is correlated with the intra-community edge given in Fig. 4a. As many intra-connected edges are cut down, the average degree reflects the decreasing nature and vice-versa. For the Cumulative Co-authorship network (Fig. 6b), the change is very trivial and remains within the range 0.5–2.5. In the Syn-1 network (Fig. 6c), most communities follow a decreasing trend of the average internal degree of the community

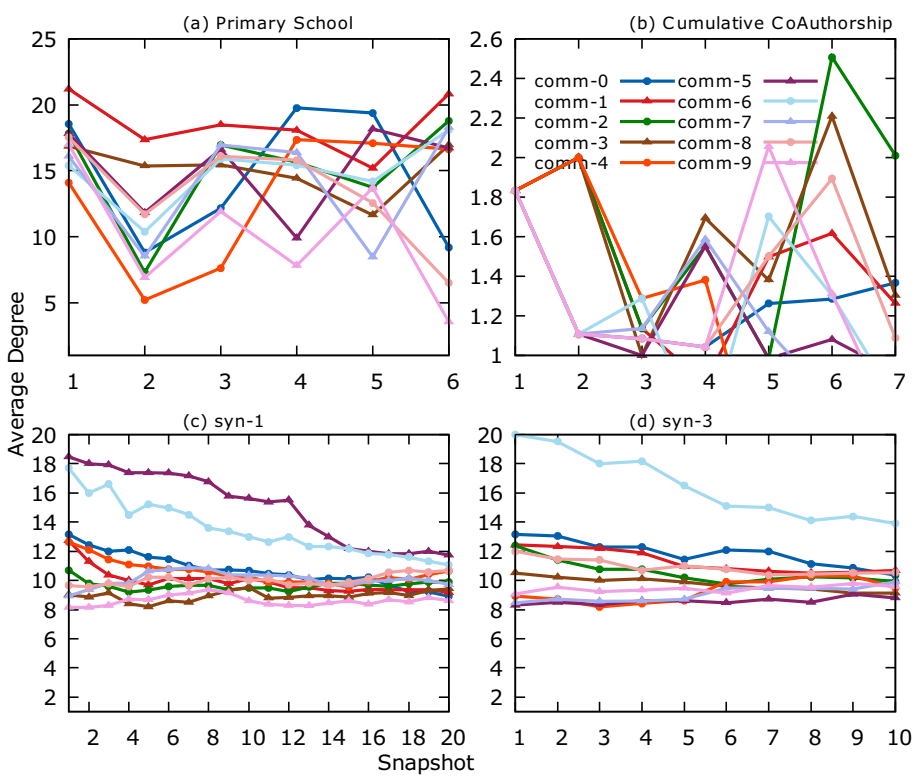

Fig. 6 Average degree values for different networks (**a**–**d**) in distinct snapshots of the networks

members over time, with a few spikes in some of the snapshots. The change is very minimal within the range of 1.3 to 2.4. The Syn-3 network (Fig. 6d) follows a similar pattern as the Syn-1 network.

FOMD

In the Primary School network (Fig. 7a), the values of FOMD for the selected communities follow an almost similar pattern as the Intra-Community Edge metric seen in Fig. 4a. For the Cumulative Co-Authorship network (Fig. 7b), most of the communities share the same values and follow a decreasing trend over the snapshots. In both Syn-1 (Fig. 7c) and Syn-3 (Fig. 7d) networks, the values of the FOMD metric vary within a very small range (0.01), close to 0.

Inter-community edge

In the Primary School network (Fig. 8a), the inter-community edges for the selected communities have a sharp increase and (decrease) in snapshots 2, 5 and (3, 6) respectively. In the Cumulative CoAuthorship network (Fig. 8b), the inter-community edges for the selected communities follow a mixed pattern including both increasing and decreasing nature throughout the snapshots. In both Syn-1 (Fig. 8c) and Syn-3 (Fig. 8d) networks, the selected communities follow a mixed trend of both increasing and decreasing patterns but very small changes in values.

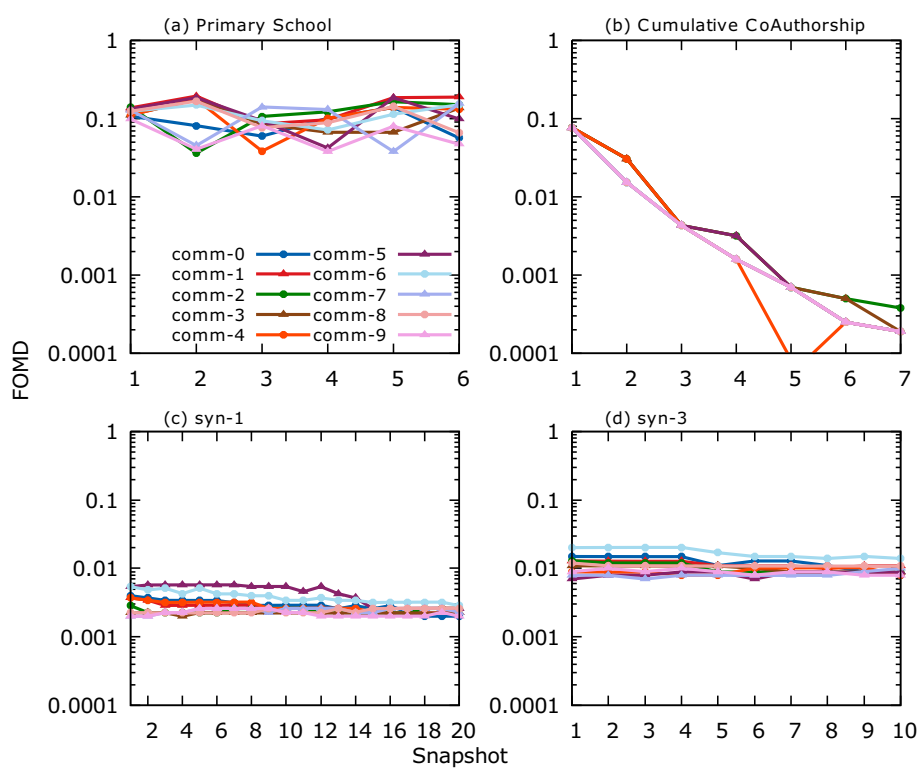

Fig. 7 FOMD values for different networks (**a**–**d**) in distinct snapshots of the networks

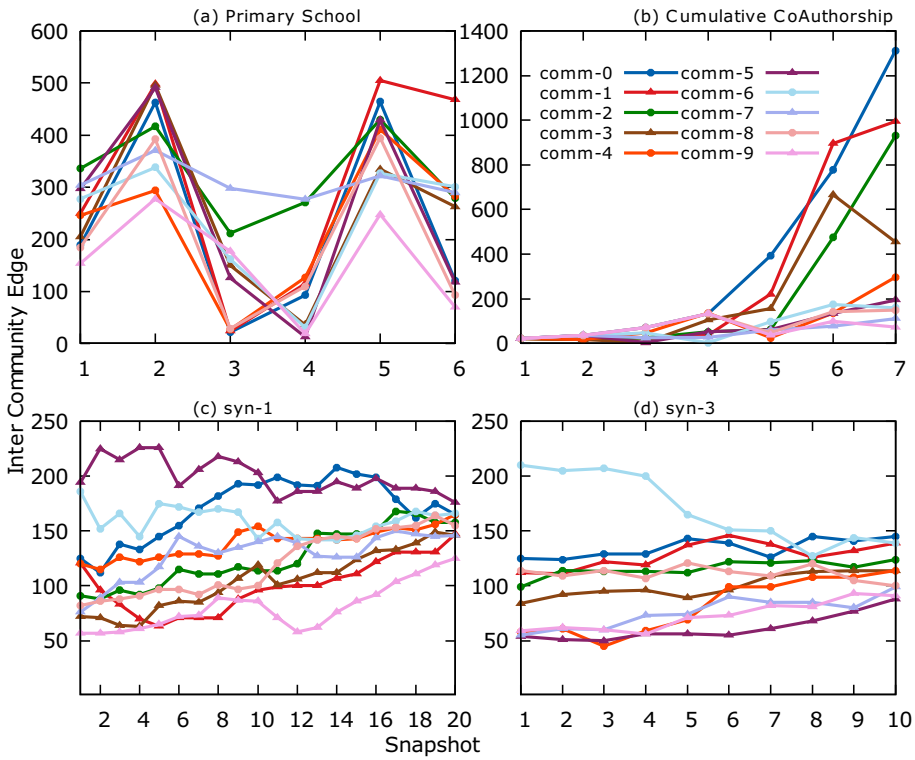

Fig. 8 Inter-community edges for different networks (**a**–**d**) in distinct snapshots of the networks

Expansion

In the Primary School network (Fig. 9a), the values indicate a positive correlation with cut ratio and normalized cut. The graph also follows a similar pattern as given in Figs. 10a and 11a. In the Cumulative CoAuthorship network (Fig. 9b), the value of expansion maintains a positive relationship with the cut ratio (Fig. 10b) and normalized cut (Fig. 11b).

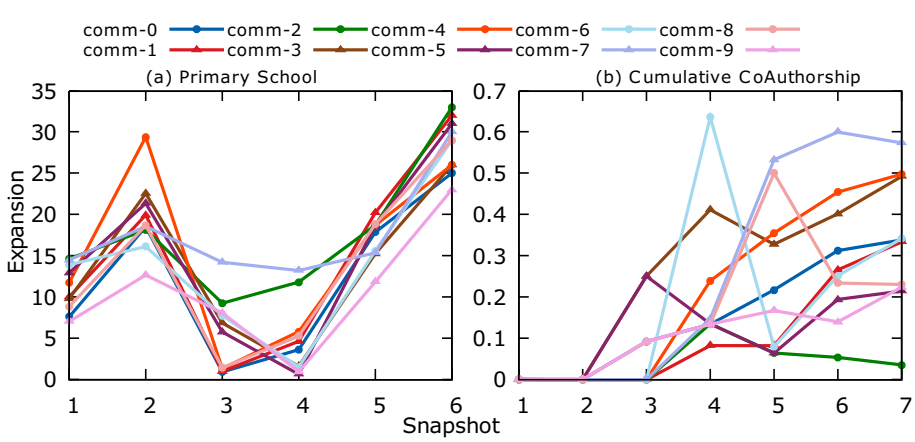

Fig. 9 Expansion values for different networks (**a**, **b**) in distinct snapshots of the networks. Both Syn-1 and Syn-3 networks have almost constant and very small values for this graph metric and are considered negligible

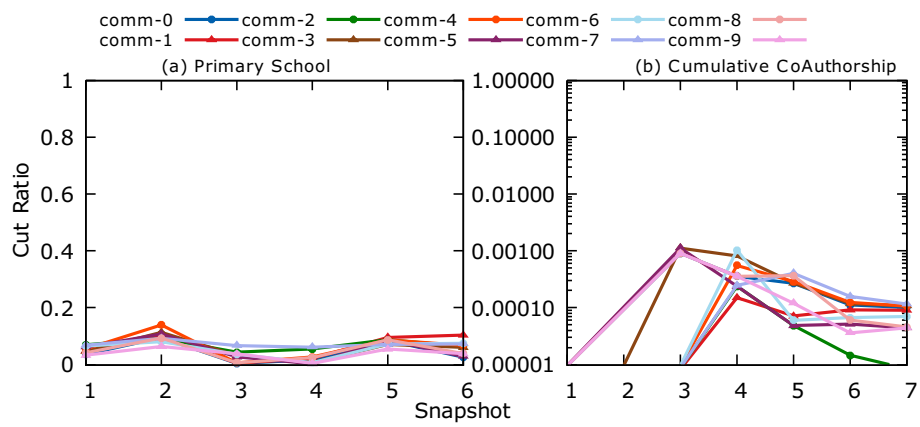

Fig. 10 Cut Ratio values for different networks (**a**, **b**) in distinct snapshots of the networks. Both Syn-1 and Syn-3 networks have almost constant and very small values for this graph metric and are considered negligible

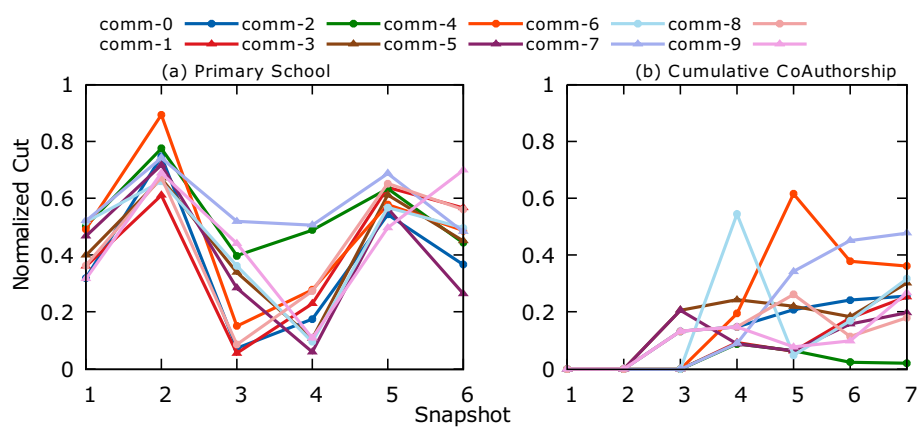

Fig. 11 Normalized Cut values for different networks (**a**, **b**) in distinct snapshots of the networks. Both Syn-1 and Syn-3 networks have almost constant and very small values for all of these metrics and are considered negligible

Cut ratio

In the Primary School network (Fig. 10a), the values indicate a positive correlation with expansion and normalized cut. The graph follows a similar pattern as given in Figs. 9a and 11a. In the Cumulative CoAuthorship network (Fig. 10b), the value of the cut ratio maintains a positive relationship with expansion (Fig. 9b) and normalized cut (Fig. 11b).

Normalized cut

For both the Primary School (Fig. 11a) and the Cumulative CoAuthorship networks (Fig. 11b), the values and graph patterns are almost the same as conductance given in Fig. 12a, b respectively.

Conductance

In the Primary School network (Fig. 12a), the values indicate a positive correlation with expansion, cut ratio, and normalized cut. The plot also follows a similar pattern as given

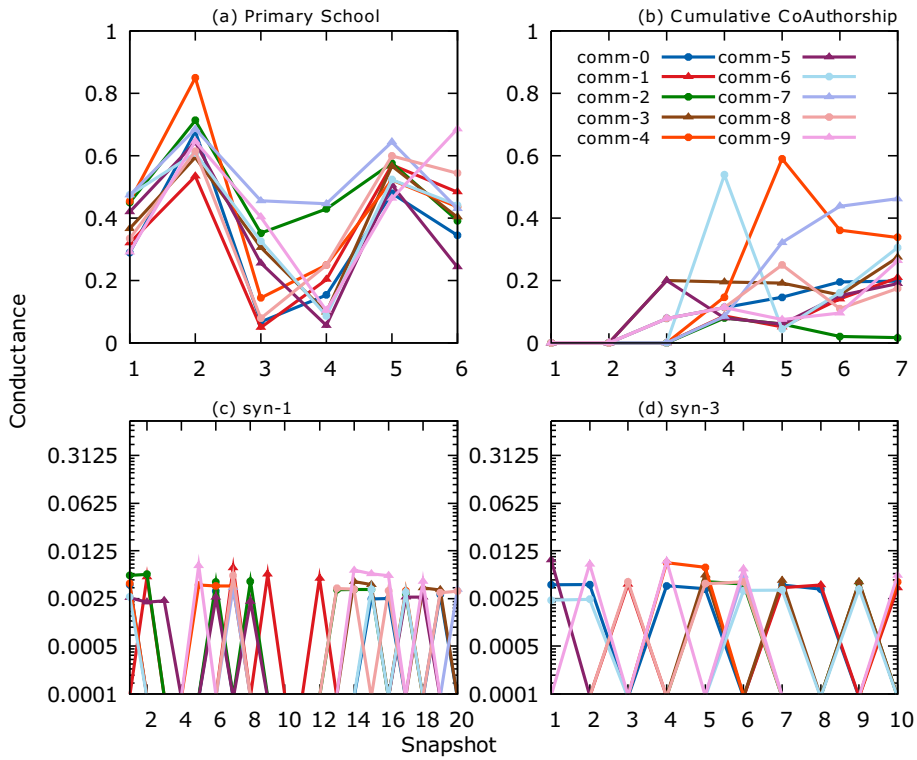

Fig. 12 Conductance values for different networks (**a**–**d**) in distinct snapshots of the networks

in Figs. 9a, 10a, and 11a. In the Cumulative CoAuthorship network (Fig. 12b), the value of conductance maintains a positive relationship with expansion (Fig. 9b), cut ratio (Fig. 10b), and normalized cut (Fig. 11b). In both Syn-1 (Fig. 12c) and Syn-3 networks (Fig. 12d), the value of conductance is very low within the range 0–0.01, very close to 0. These small values indicate very strong connectivity among the members of the communities. It also shows the inverse relation with permanence given in Fig. 15c, d.

Clustering coefficient

In the Primary School network (Fig. 13a), the selected communities mostly follow the opposite pattern of expansion (Fig. 9a)/cut ratio (Fig. 10a) /normalized cut (Fig. 11a)/ conductance (Fig. 12a) metrics as expected by the definition. We notice a few exceptions for communities with id 4, 5. In the Cumulative CoAuthorship network (Fig. 13b), the values of the clustering coefficient follow an inverse relation with conductance (Fig. 12b). In both Syn-1 (Fig. 13c) and Syn-3 networks (Fig. 13d), the values of the clustering coefficient follow a decreasing trend throughout the snapshots.

Separability

In the Primary School network (Fig. 14a), the positive correlation with permanence (Fig. 15a) and an inverse correlation with expansion (Fig. 9a), cut ratio (Fig. 10a), normalized cut (Fig. 11a) and conductance (Fig. 12a) is maintained well. In the Cumulative CoAuthorship network (Fig. 14b), positive correlation with permanence (Fig. 15b) is maintained except for a few exceptions (comm-id 6).

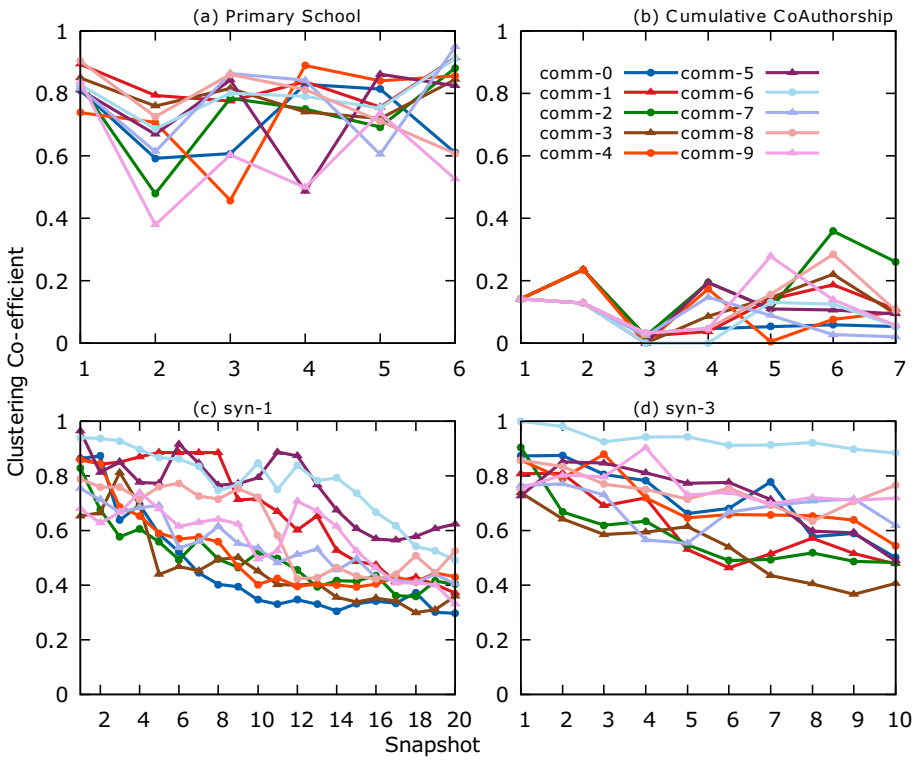

Fig. 13 Clustering coefficient values for different networks (**a**–**d**) in distinct snapshots of the networks

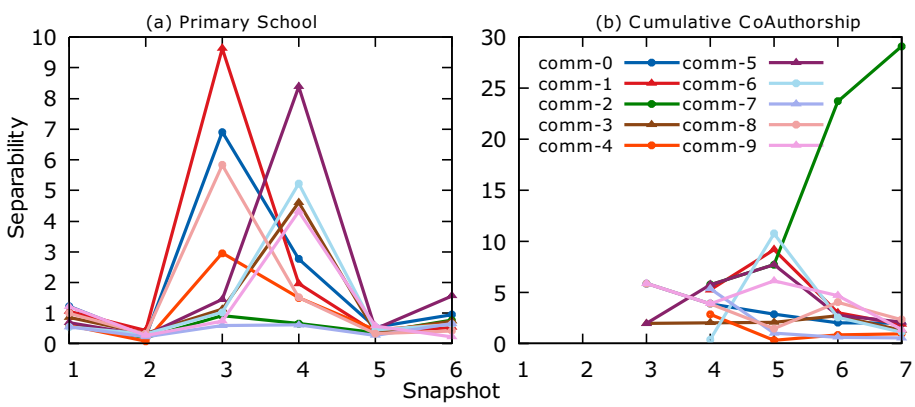

Fig. 14 Separability values for different networks (**a**, **b**) in distinct snapshots of the networks. Intra-Community Edge being 0 for most communities in the Syn-1 and the Syn-3 networks, the value of separability also becomes 0 for those communities

Permanence

In the Primary School network (Fig. 15a), the selected communities mostly follow the opposite pattern of expansion (Fig. 9a)/cut ratio (Fig. 10a) /normalized cut (Fig. 11a) metrics as expected by the definition. In the Cumulative CoAuthorship network (Fig. 15b), the selected communities follow a mixed trend. If we follow some particular communities, the relation with the other graph metrics becomes prominent. For communities with ids 2, 4 and 6, we observe that these communities have a sharp increase,

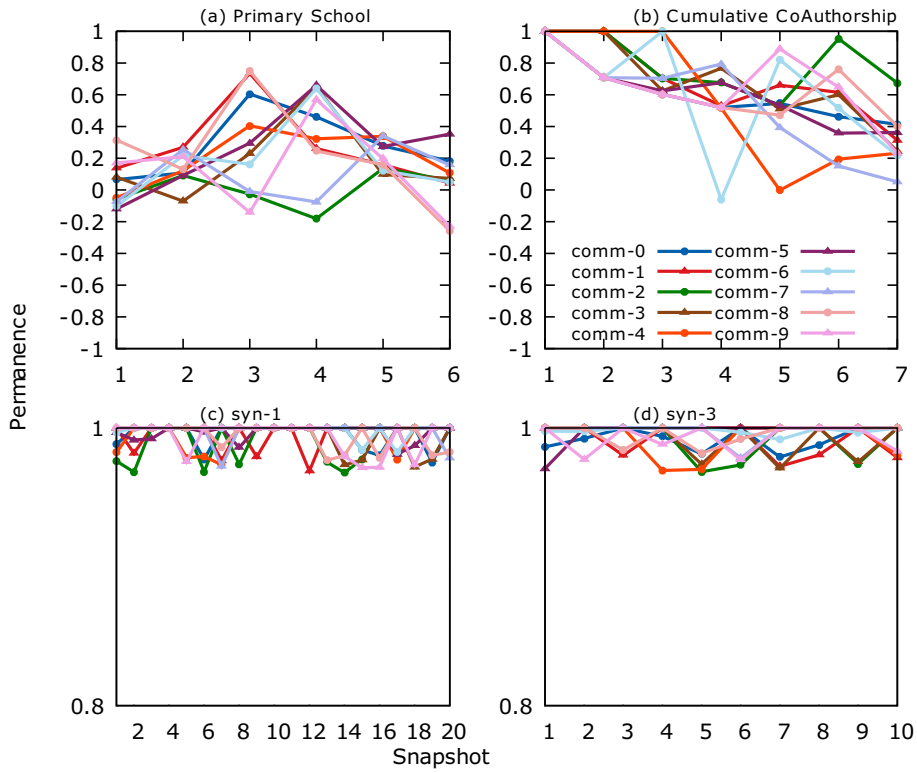

Fig. 15 Permanence values for different networks (**a**–**d**) in distinct snapshots of the networks

decrease and decrease at snapshots 6, 5, and 4 respectively. We can observe from the values of conductance (Fig. 12b) that these same communities follow just the opposite pattern. In both Syn-1 (Fig. 15c) and Syn-3 networks (Fig. 15d), the value of permanence is very high within the range 0.97–1. This indicates very strong connectivity among the members of the communities.

After analyzing all 12 graph metrics to understand the community evolution for the ground truth communities, aside from the metrics definition, experimentally we also observe that the metrics using both internal and external connections portray a better concept of community qualities. So, normalized cut, conductance, and permanence take a major role in providing more valuable insights about the community structure. In some cases, the clustering coefficient is also helpful but shows some exceptions for some of the networks. Based on this observation, we use these four graph metrics to evaluate the DCD algorithms in "Comparison of different dynamic community detection methods" section.

Comparison of different dynamic community detection methods

We have conducted a comparison of different dynamic community detection methods to observe how community structures change over time using different techniques. We use various graph metrics to understand the quality of the community structure and explain its changes. Initially, we begin with six dynamic community detection methods for comparison of the community structures throughout the snapshots. However, two of the methods, EstCon and TraN, show increasingly longer run times compared to the

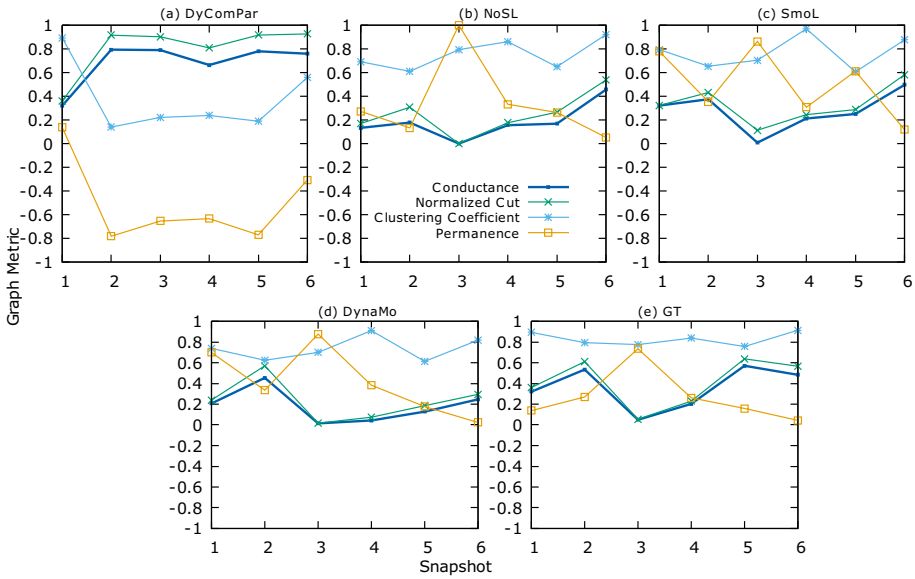

Fig. 16 Four different DCD methods (**a**–**d**), compared with Ground Truth (**e**), for a particular community in Primary School Network based on the graph metrics: *Conductance*, *Normalized Cut*, *Modularity*, and *Permanence*

Table 7 Runtime of 6 different dynamic community detection algorithms for different networks

Network	DyPerm (s)	NoSL (s)	SmoL (s)	DynaMo (s)	EstCon (h)	TraN (h)
Primary School	0.61	6.50	9.30	0.55	10.01	1.09
CollegeMsg	10.10	28.40	100.51	8.21	22.29	12.83
fb-forum	88.15	297.10	151.92	92.73	–	–
citHepTh	16,896.09	18,751.00	20,900.47	18,551.91	–	–
Cumulative CoAuthorship	3765.90	2971.60	4011.66	4197.10	–	–
Non-cumulative CoAuthorship	3525.50	8193.70	6957.10	4212.97	–	–
syn-1	677.26	1889.54	1511.64	507.94	–	–
syn-2	2258.98	5076.01	3549.30	2713.04	–	–
syn-3	218.47	507.72	392.81	220.24	–	–

other four methods in Table 7. Therefore, we eliminate these two methods from further comparisons.

From Figs. 16, 17, 18, and 19, we observe that the community quality is similar as indicated by the graph metric values. To simplify the results, we present the graph metrics: Conductance, Normalized Cut, Clustering Coefficient, and Permanence, which consider both internal and external connectivity. We also observe a close correlation among these four metrics from Figs. 16, 17, 18, and 19. Conductance and Normalized Cut are inversely correlated to Clustering Coefficient, and Permanence in most cases. We also consider the algorithms' runtime given in Table 7. We find that both DynaMo and DyPerm have the least processing time, but we prioritize the "Permanence" metric over "Modularity" because it involves local optimization compared to global optimization. Again, the order in which edges are added or deleted has no impact on the permanence maximization (Agarwal et al. 2018). But in modularity optimization, the order of vertex

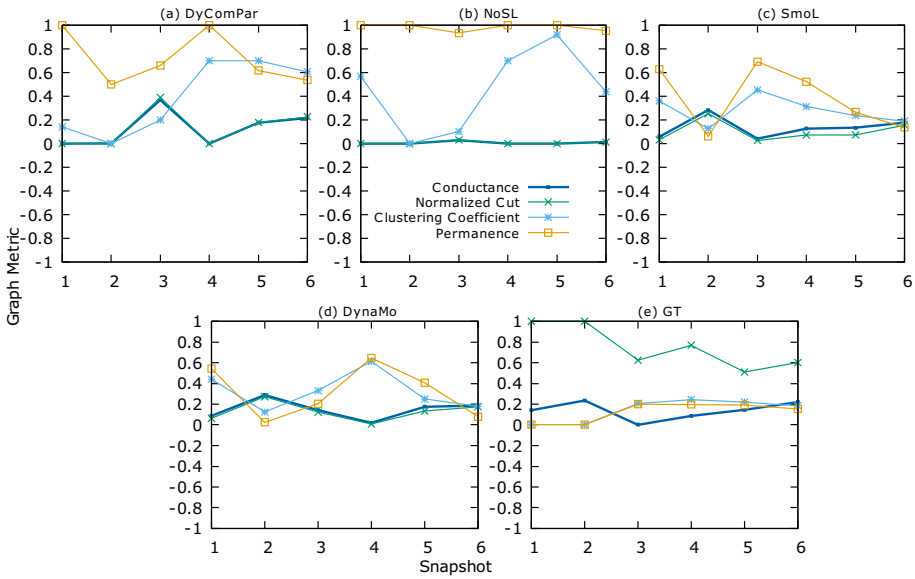

Fig. 17 Four different DCD methods (**a**–**d**), compared with Ground Truth (**e**), for a particular community in Cumulative CoAuthorship Network based on the graph metrics: *Conductance*, *Normalized Cut*, *Modularity*, and *Permanence*

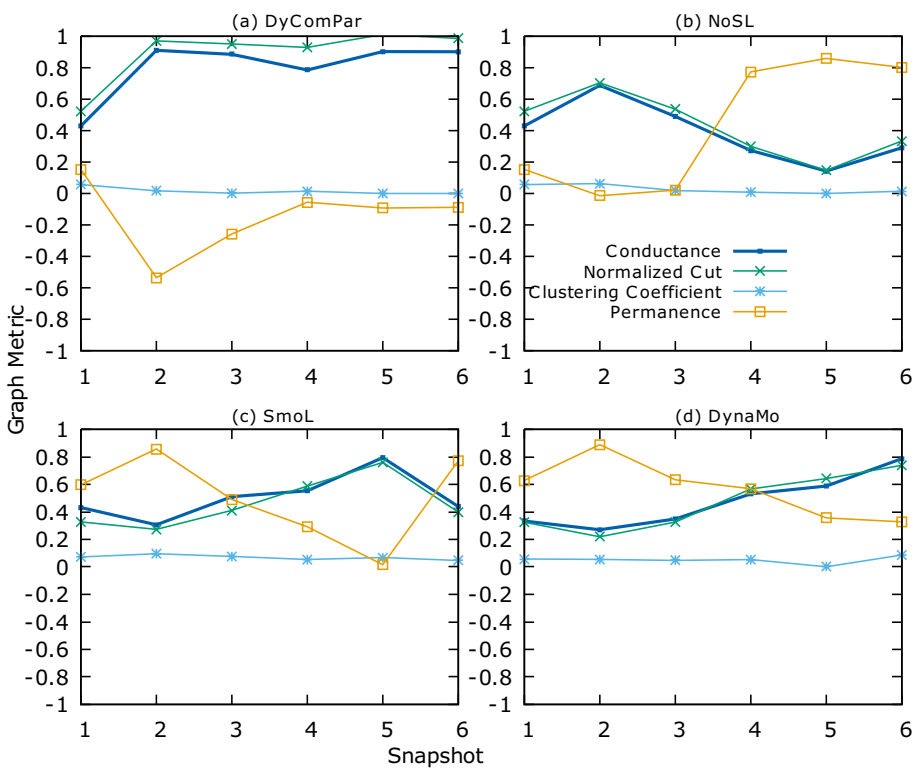

Fig. 18 Comparison of four different DCD methods (**a**–**d**) for a particular community in College Message Network based on the graph metrics: *Conductance*, *Normalized Cut*, *Modularity*, and *Permanence*. Ground Truth communities are not available for this dataset

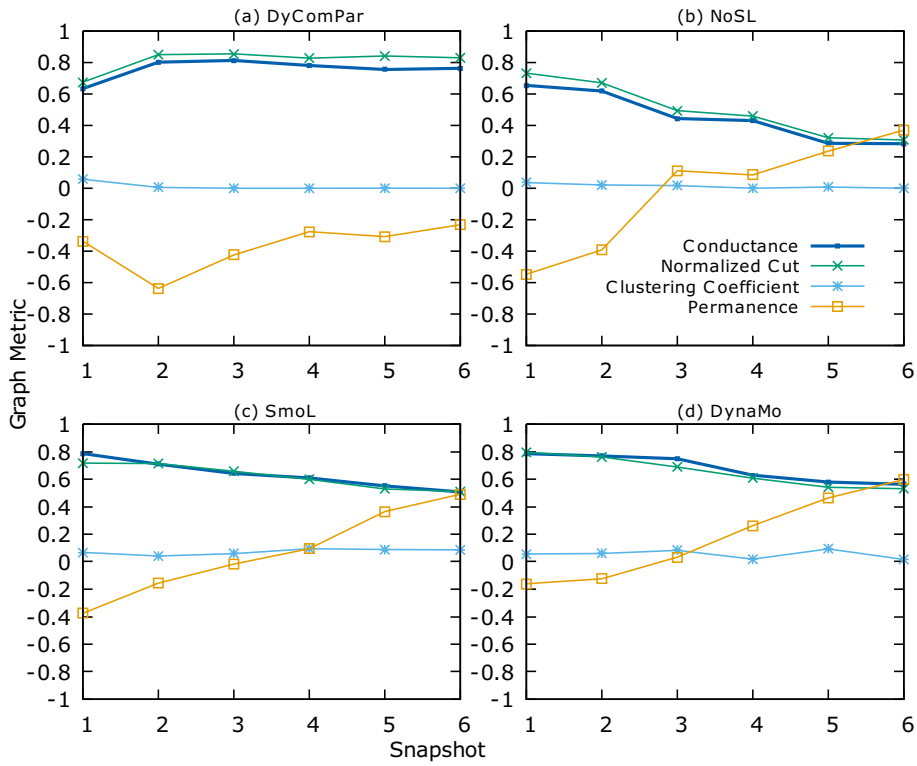

Fig. 19 Comparison of four different DCD methods (**a**–**d**) for a particular community in Facebook Forum Network based on the graph metrics: *Conductance*, *Normalized Cut*, *Modularity*, and *Permanence*. Ground Truth communities are not available for this dataset

changes the convergence, and randomness is introduced depending on the order. Both of these make "Permanence" a good choice in designing a parallel algorithm.

Parallel performance: *DyComPar*

We evaluate the performance of *DyComPar* in terms of speed-up, which is a measure of how much faster the parallel implementation is compared to the serial implementation.

Figure 20 shows the speed-up achieved by *DyComPar* for different types of networks. The speed-up values range from 4× to 18×, indicating that *DyComPar* performs significantly faster than the serial implementation, DyPerm. Specifically, the speed-up of 4× to 18× implies that *DyComPar* can process the input data in a fraction of the time required by the serial implementation, DyPerm, which is a considerable performance improvement. Overall, the parallel performance of *DyComPar* suggests that the algorithm can be an effective solution for detecting dynamic communities in large-scale networks, particularly for networks with a high degree of complexity and dynamic behavior.

Discussion

This study provides valuable insights to researchers seeking to explore various community detection methods for dynamic networks. They can evaluate the performance of each algorithm by analyzing the quality of the community structures and the effectiveness of all DCD algorithms. To enhance the design of a parallel algorithm, we leverage

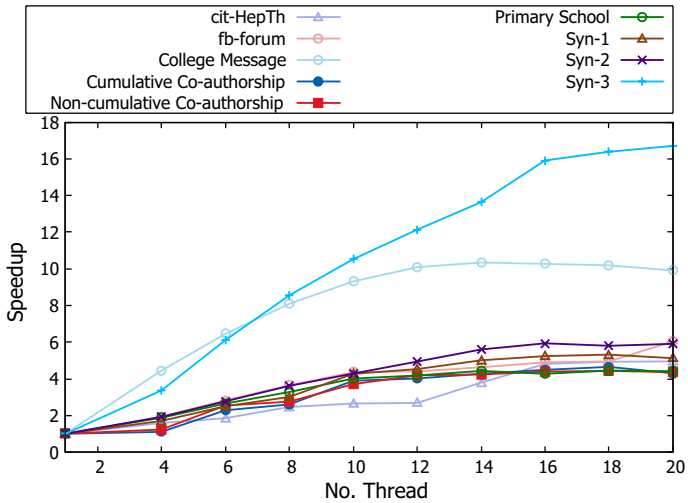

Fig. 20 Speed-up for shared-memory parallel algorithm *DyComPar* for different networks with a varied number of threads

the benefits of completing more independent tasks in parallel using a local optimization metric. Thus, we developed our parallel algorithm using the vertex-based local optimization metric of permanence. Additionally, we examine the evolution of communities in various snapshots to identify any relationship between permanence and other graph metrics that could optimize our parallel algorithm. We discovered that normalized cut and conductance metrics demonstrate an inverse relationship with permanence across most of the networks used in our experiments. However, both of these metrics require almost similar or more computing complexity than permanence calculation. Furthermore, due to the distinct characteristics of networks from different domains, establishing a universal relationship with other simple metrics is challenging. As a result, we did not incorporate any optimization into our algorithm based on these observed metrics. In summary, our study contributes to the understanding of community evolution and quality by analyzing multiple graph metrics and different dynamic community detection methods. We provide a scalable parallel algorithm, *DyComPar*, for dynamic community detection.

Conclusion

In this work, we focus on investigating how community structures change over time in various real-world dynamic networks. We analyze the evolution of ground truth communities across snapshots using different graph metrics that depend on the internal, external, or both (internal and external) connectivity among community members. We conduct experiments on several real-world and synthetic networks from different domains and evaluate six state-of-the-art dynamic community detection (DCD) methods. We verify the quality of the community structures derived from these methods with respect to ground truth. We observe that for certain networks, the quality of the community is almost similar using different methods. Our experiments reveal that runtime plays a significant role in choosing the algorithms for detecting dynamic communities. DynaMo and DyPerm are comparable to each other in terms of runtime, and

we prioritize choosing the "permanence" metric over "modularity" to emphasize local optimization rather than global optimization for an efficient parallel algorithm design. Our study will be useful for the research community to quickly learn about the derived output community quality or structure found from different DCD algorithms. Based on our findings, we choose to implement a parallel dynamic community detection algorithm based on DyPerm due to the advantages it provides as a vertex-centric metric. We implement a shared-memory parallel algorithm called *DyComPar* using multi-threading and loop parallelization. Our results show a 4–18 fold speed-up on different networks. Future research includes extending and designing a distributed parallel DCD algorithm using the local optimization benefit of permanence.

Author contributions
NS, AB, KZI, and SA proposed the methods. NS implemented the code and performed the experiments. NS and SA contributed to writing the manuscript. All authors read and approved the final manuscript.

Funding
This work has been supported by a National Science Foundation grant (Award No. 2323533). The work is also partially supported by Lawrence Berkeley National Laboratory under Contract No. DE-AC02-05CH11231 with the U.S. Department of Energy and the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Dept. Of Energy and National Nuclear Security Administration.

Availability of data and materials
All data generated or analyzed during this study are included in this published article

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 30 May 2023 Accepted: 7 September 2023
Published online: 18 September 2023

References

Agapito G, Milano M, Cannataro M (2022) Parallel network analysis and communities detection (PANC) pipeline for the analysis and visualization of covid-19 data. Parallel Process Lett 32(01n02):2142002
Agarwal P, Verma R, Agarwal A, Chakraborty T (2018) Dyperm: Maximizing permanence for dynamic community detection. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 437–449
Ammar K (2023) Systems and algorithms for dynamic graph processing. University of Waterloo
Badlani R, Culberg K, Jiang Z (2018) Community detection and evolution in temporal networks. CS224W Analysis of Networks MINING AND LEARNING WITH GRAPHS Project Report Autumn 2018 https://snap.stanford.edu/class/cs224w-2018/projects.html. http://snap.stanford.edu/class/cs224w-2018/reports/CS224W-2018-50.pdf
Bautista E, Latapy M (2023) A frequency-structure approach for link stream analysis. In: Temporal network theory, 2nd edn. https://hal.science/hal-04086777
Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):10008
Cazabet R, Boudebza S, Rossetti G (2020) Evaluating community detection algorithms for progressively evolving graphs. J Complex Netw 8(6):027
Chakrabort T, Sikdar S, Tammana V, Ganguly N, Mukherjee A (2013) Computer science fields as ground-truth communities: their impact, rise and fall. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining, pp 426–433
Chakraborty T, Dalmia A, Mukherjee A, Ganguly N (2017) Metrics for community analysis: a survey. ACM Comput Surv (CSUR) 50(4):1–37
Chakraborty T, Srinivasan S, Ganguly N, Mukherjee A, Bhowmick S (2014) On the permanence of vertices in network communities. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1396–1405
DATASETS. http://www.sociopatterns.org/datasets

Dilmaghani S, Brust MR, Ribeiro CH, Kieffer E, Danoy G, Bouvry P (2022) From communities to protein complexes: a local community detection algorithm on PPI networks. PLoS ONE 17(1):0260484
Documentation | User Guides | QB2. http://www.hpc.lsu.edu/docs/guides.php? system=QB2
Duan D, Li Y, Jin Y, Lu Z (2009) Community mining on dynamic weighted directed graphs. In: Proceedings of the 1st ACM international workshop on complex networks meet information & knowledge management. CNIKM '09, pp. 11–18. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1651274.1651278
Fang F, Wang T, Tan S, Chen S, Zhou T, Zhang W, Guo Q, Liu J, Holme P, Lu X (2022) Network structure and community evolution online: behavioral and emotional changes in response to covid-19. Front Public Health 9:813234
Feng G, Meng X, Ammar K (2015) Distinger: a distributed graph data structure for massive dynamic graph processing. In: 2015 IEEE international conference on big data (big data). IEEE, pp 1814–1822
Gabert K, Pinar A, Çatalyürek ÜV (2021) Shared-memory scalable k-core maintenance on dynamic graphs and hypergraphs. In: 2021 IEEE international parallel and distributed processing symposium workshops (IPDPSW. IEEE), pp 998–1007
Gabert K, Sancak K, Özkaya MY, Pinar A, Çatalyürek ÜV (2021) Elga: elastic and scalable dynamic graph analysis. In: Proceedings of the international conference for high performance computing, networking, storage and analysis, pp 1–15
Gemmetto V, Barrat A, Cattuto C (2014) Mitigation of infectious disease at school: targeted class closure vs school closure. BMC Infect Dis 14(1):695. https://doi.org/10.1186/PREACCEPT-6851518521414365
Girvan M, Newman ME (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826
Green O, Bader DA (2016) custinger: supporting dynamic graph algorithms for GPUS. In: 2016 IEEE high performance extreme computing conference (HPEC). IEEE, pp 1–6
Guo C, Wang J, Zhang Z (2014) Evolutionary community structure discovery in dynamic weighted networks. Physica A 413:565–576
Halappanavar M, Lu H, Kalyanaraman A, Tumeo A (2017) Scalable static and dynamic community detection using grappolo. In: 2017 IEEE high performance extreme computing conference (HPEC). IEEE, pp 1–6
Karimi F, Lotfi S, Izadkhah H (2020) Multiplex community detection in complex networks using an evolutionary approach. Expert Syst Appl 146:113184
Kawadia V, Sreenivasan S (2012) Sequential detection of temporal communities by estrangement confinement. Sci Rep 2(1):1–10
Kazemzadeh F, Safaei AA, Mirzarezaee M (2022) Influence maximization in social networks using effective community detection. Physica A 598:127314
Khanda A, Srinivasan S, Bhowmick S, Norris B, Das SK (2021) A parallel algorithm template for updating single-source shortest paths in large-scale dynamic networks. IEEE Trans Parallel Distrib Syst 33(4):929–940
Lancichinetti A, Fortunato S (2009) Community detection algorithms: a comparative analysis. Phys Rev E 80(5):056117
Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys Rev E 78(4):046110
Leskovec J, Krevl A (2014) SNAP datasets: stanford large network dataset collection. http://snap.stanford.edu/data
Li T, Wang W, Wu X, Wu H, Jiao P, Yu Y (2020) Exploring the transition behavior of nodes in temporal networks based on dynamic community detection. Futur Gener Comput Syst 107:458–468
Liu F, Wu J, Xue S, Zhou C, Yang J, Sheng Q (2020) Detecting the evolving community structure in dynamic social networks. World Wide Web 23:715–733
Martinet L-E, Kramer M, Viles W, Perkins L, Spencer E, Chu C, Cash S, Kolaczyk E (2020) Robust dynamic community detection with applications to human brain functional networks. Nat Commun 11(1):2785
Mucha PJ, Richardson T, Macon K, Porter MA, Onnela J-P (2010) Community structure in time-dependent, multiscale, and multiplex networks. Science 328(5980):876–878
Naik D, Ramesh D, Gandomi AH, Gorojanam NB (2022) Parallel and distributed paradigms for community detection in social networks: A methodological review. Expert Syst Appl 187:115956
Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113
Pandey P, Wheatman B, Xu H, Buluc A (2021) Terrace: a hierarchical graph container for skewed dynamic graphs. In: Proceedings of the 2021 international conference on management of data, pp 1372–1385
Peixoto TP, Rosvall M (2019) Modelling temporal networks with Markov chains, community structures and change points. Temporal Netw Theory 65–81
Pereira FS, Gama J, Amo S, Oliveira GM (2018) On analyzing user preference dynamics with temporal social networks. Mach Learn 107(11):1745–1773
Qiao S, Han N, Gao Y, Li R-H, Huang J, Sun H, Wu X (2020) Dynamic community evolution analysis framework for large-scale complex networks based on strong and weak events. IEEE Trans Syst Man Cybern Syst 51(10):6229–6243
Rajita B, Shukla M, Kumari D, Panda S (2021) An experimental analysis of community detection algorithms on a temporally evolving dataset. In: Recommender systems. CRC Press, pp. 23–70
Rossi RA, Ahmed NK (2015) The network data repository with interactive graph analytics and visualization. In: AAAI. https://networkrepository.com
Sarıyüce AE, Gedik B, Jacques-Silva G, Wu K-L, Çatalyürek ÜV (2016) Sonic: streaming overlapping community detection. Data Min Knowl Disc 30:819–847
Sattar NS, Arifuzzaman S (2018) Overcoming MPI communication overhead for distributed community detection. In: Workshop on software challenges to exascale computing. Springer, pp. 77–90
Sattar NS, Arifuzzaman S (2018) Parallelizing louvain algorithm: distributed memory challenges. In: 2018 IEEE 16th international conference on dependable, autonomic and secure computing, 16th international conference on pervasive intelligence and computing, 4th intl conf on big data intelligence and computing and cyber science and technology congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, pp 695–701
Sattar NS, Arifuzzaman S (2020) Community detection using semi-supervised learning with graph convolutional network on gpus. In: 2020 IEEE international conference on big data (big data). IEEE, pp 5237–5246

Sattar NS, Arifuzzaman S (2022) Scalable distributed Louvain algorithm for community detection in large graphs. J Supercomput 78:10275–10309
Sattar NS (2019) Scalable community detection using distributed Louvain algorithm. Master's thesis, University of New Orleans, Computer Science Department. https://scholarworks.uno.edu/td/2640/
Sattar NS (2021) Parallel algorithms for scalable graph mining: Applications on big data and machine learning. In: Doctoral showcase, 2021 international conference for high performance computing, networking, storage, and analysis (SC'21). https://sc21.supercomputing.org/proceedings/doctoral_showcase/doc_showcase_pages/drs111.html
Sattar NS (2022) Parallel algorithms for scalable graph mining: applications on big data and machine learning. Ph.D. Dissertation, University of New Orleans, Computer Science Department. https://scholarworks.uno.edu/td/3014/
Stehlé J, Voirin N, Barrat A, Cattuto C, Isella L, Pinton J, Quaggiotto M, Van den Broeck W, Régis C, Lina B, Vanhems P (2011) High-resolution measurements of face-to-face contact patterns in a primary school. PLOS ONE 6(8):23176. https://doi.org/10.1371/journal.pone.0023176
Wheatman B, Xu H (2018) Packed compressed sparse row: a dynamic graph representation. In: 2018 IEEE high performance extreme computing conference (HPEC). IEEE, pp 1–7
Yang J, Leskovec J (2015) Defining and evaluating network communities based on ground-truth. Knowl Inf Syst 42(1):181–213
Zhang C, Zhang Y, Wu B (2018) A parallel community detection algorithm based on incremental clustering in dynamic network. In: 2018 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). IEEE, pp 946–953
Zhuang D, Chang MJ, Li M (2019) Dynamo: dynamic community detection by incrementally maximizing modularity. IEEE Trans Knowl Data Eng 33(5):1934–1945
Zou L, Zhang F, Lin Y, Yu Y (2023) An efficient data structure for dynamic graph on GPUS. IEEE Trans Knowl Data Eng

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com