A comparative study of community detection techniques for large evolving graphs

Lauranne Coppens, Jonathan De Venter, Sandra Mitrović[0000−0002−5697−5865], and Jochen De Weerdt[0000−0001−6151−0504]

Research Center for Information Systems Engineering (LIRIS)
KU Leuven, Leuven, Belgium
sandra.mitrovic@kuleuven.be

Abstract. Community detection has recently received increased attention due to its wide range of applications in many fields. While at first most techniques were focused on discovering communities in static networks, lately the focus has shifted toward evolving networks because of their high relevance in real-life problems. Given the increasing number of the methods being proposed, this paper explores the current availability of empirical comparative studies of dynamic methods and also provides its own qualitative and quantitative comparison with the aim of gaining more insight in the performance of available methods. The results show that no single best performing community detection technique exists, but rather, the choice of the method depends on the objective and dataset characteristics.

Keywords: Dynamic Community Detection · Large Evolving Graphs · RDyn.

1 Introduction

Community detection techniques in complex networks are a well-covered topic in academic literature nowadays as identifying meaningful substructures in complex networks has numerous applications in a vast variety of fields ranging from biology, mathematics, and computer science to finance, economics and sociology. A majority of the literature covers static community detection algorithms, i.e. algorithms used to uncover communities in static networks. However, real-world networks often possess temporal properties as nodes and edges can appear and disappear, potentially resulting in a changed community structure. Consequently, researchers have recently taken a keen interest in community detection algorithms that can tackle dynamic networks. Given the increasing number of the methods being proposed, a systematic comparison of both their algorithmic and performance differences is required so as to be able to select a suitable method for a particular community discovery problem. Nonetheless, newly proposed community detection methods for dynamic graphs are typically compared with only very few methods in settings aiming to demonstrate superiority of the proposed method. Consequently, the setup and results of these comparisons might contain an unconscious bias towards one's own algorithm. As such, a well-founded and extensive comparative analysis of dynamic community detection (DCD) techniques is missing in the current literature. This is not surprising given the many different aspects which come into play when comparing DCD

methods: different underlying network models, different community definitions, different temporal segments used for detecting communities, different community evolution events tracked etc.

To bridge this literature gap, we perform a qualitative and quantitative comparison of DCD techniques. To this end, we adopt the classification system of Cazabet and Rossetti [1] to provide a concise framework within which the comparison is framed. For our qualitative comparison, we focus on relevant community detection characteristics like community definition used, ability to detect different type of communities and community evolution events as well as computational complexity. For quantitative analysis we report computational time and partition quality in terms of NF1 statistics, on 900 synthetic RDyn [7] and one real-world DBLP dataset [25]. Results showcase that no single best performing community technique exists. Instead, the choice of the method should adapt to the dataset and the final objective.

2 Methodology for Unbiased Comparison of Dynamic Community Detection Methods

In this section we provide details of our comparative study which basically consists of three parts: first, shortlisting candidate algorithms to be compared, second, analyzing their algorithmic characteristics and, third, performing the empirical analysis.

2.1 Algorithm selection

Given the soundness and completeness of Cazabet and Rossetti's classification framework [1], we opt for using this framework as a steering wheel in the process of method selection. Within this framework, three large types of dynamic algorithms for searching communities are distinguished: 1) those that only consider the current state of the network (instant-optimal); 2) those that only consider past and present clustering and past instances of the network topology (temporal trade-off); 3) those that consider the entire network evolution available in the data, both past and future clustering (cross-time).

In an applied setting, neglecting previous states of the network oftentimes leads to sub-optimal solutions. Additionally, it is realistic to assume that communities will be updated using data that become available periodically. Consequently, no future information will be available at time t. With this in mind, we opt for focusing on temporal trade-off algorithms. Within Cazabet and Rossetti's framework, these are further subdivided into four categories: Global Optimization (denoted originally as 2.1), Rule-Based Updates (2.2), Multi-Objective Optimization (2.3) and Network Smoothing (2.4). For each subcategory, at least one representative is chosen. Additionally, the list of compared algorithms is complemented by more recently published techniques, which, in turn, are also classified in the four previously mentioned categories.

Moreover, three characteristics are instrumental in the selection. Firstly, the algorithm has to be able to detect communities in evolving graphs. Secondly, the algorithm would preferably be able to detect overlapping communities to ensure a realistic partitioning in social network problems. Thirdly, the capability of extracting community

evolution events is a desired trait with the goal of having realistic partitions that incorporate as much available information as possible in the partitioning process. Finally, some algorithms will be included as benchmark algorithms in order to compare results with previously performed comparative analyses.

2.2 Qualitative analysis

The qualitative analysis is based on the comparison of algorithmic characteristics. In particular, comparison is performed with respect to the following six questions:

1. How does the method search for communities? In other words, which of the categories within the framework of Cazabet and Rossetti does it fit in (if any)?
2. What community definition is adopted (modularity, density, conductance...)?
3. How efficient the method is? That is, what is its computational complexity?
4. Which community evolution events can the method track (birth, death, merge, split, growth, contraction, continuation, resurgence)?
5. Can the method find overlapping communities?
6. Can the method find hierarchical communities?

2.3 Empirical analysis setup

Given their different characteristics, to provide a fair comparison, selected DCD methods are benchmarked based on both synthetic and real-life datasets. As synthetic datasets, 9 different RDyn graphs [7] were created by varying the number of nodes to 1000, 2000 and 4000 and the communities size distribution parameter α to be 2.5, 3 and 3.5. Larger α makes the sizes of communities relatively larger, more dispersed, while smaller makes the differences between community sizes smaller, more uniform. The rate of node appearance and vanishing is fixed to 0.05 and 0.02 respectively. The appearance rate is slightly larger than the vanishing rate in an attempt to mimic a slowly growing graph which could resemble, for instance, a customer base where customers enter, remain for a (long) while, and churn. For each of the 9 different RDyn graphs, 100 RDyn instances are created, yielding 900 graphs in total. The specific number 100 was arbitrarily chosen but is used to account for variations in the results.

As for the real-life dataset, the co-authorship graph from [25] was used. This dataset was originally extracted from DBLP database and for purposes of this analysis, was further limited to data from 1971 to 2002. Resulting dataset has 850 875 nodes, which represent 303 628 unique authors, and 1 656 873 edges.

To measure the relative performance of the different algorithms, two metrics were chosen. On the one side, the quality of the partition is measured by Normalized F1-statistics (NF1) and on the other side, the efficiency of the algorithm is reported in terms of the computation time.

3 Results

In this section we provide results of each of the three phases of our comparative analysis: algorithm selection, qualitative and quantitative analysis.

3.1 Algorithm selection

For the broad selection, the initial list of 51 papers on DCD methods was used [6, 13–18, 20–23, 27–65]. It was obtained by supplementing 32 temporal trade-off algorithms [6, 13–15, 17, 21, 22, 24, 27–50] from [1] with 19 algorithms not included in the aforementioned survey [16, 18, 20, 23, 51–65] that nonetheless possess interesting characteristics with regards to community and evolution extraction. Figure 1 illustrates the relevance of adding those 19 papers as it ensures the inclusion of more recent methods.

After this list of algorithms was compiled, the three algorithm-specific characteristics mentioned before were compared in order to select the approximately ten most promising algorithms that will be compared qualitatively and empirically. Following the analysis mentioned above, 13 algorithms were selected to be compared, as follows.

Partition update by global optimization (2.1) This category contains algorithms that incrementally update and find communities by globally optimizing a metric such as modularity, density or other utility functions. Two methods represent this category in the analysis. Firstly, D-GT is a game-theory based algorithm proposed in [13] for dynamic social networks. The technique considers nodes as rational agents, maximizes a utility function and finds the optimal structure when a Nash equilibrium is reached. Secondly, Updated BGL is a modularity based incremental algorithm designed by [14]. It is more time-efficient than its modularity-based peers that do not rely on community updating. Both D-GT and Updated BGL are capable of tracking community evolution events.

Partition update by set of rules (2.2) This category seems to be the most promising in terms of efficiency and accuracy for algorithms that take into account past information. The algorithms belonging to this category all consider a list of historic network changes in order to update the network's partitioning. AFOCS is an algorithm designed for performing well in mobile networks, such as online social networks, wireless sensor networks and Mobile Ad Hoc Networks [15]. The technique is able to uncover overlapping communities in an efficient way by incrementally updating the communities based on past information. It avoids the recalculation of communities at each time step, by identifying community evolution events based on four network changes, namely the appearance of a new node or edge and the removal of an edge or node. The algorithm applies different rules on how to update communities depending on which events occur. HOCTracker is a technique designed to detect hierarchical and overlapping communities in online social networks [16]. The approach detects community evolution by comparing significant evolutionary changes between consecutive time steps, reducing the number of operations to be performed by the algorithm. The algorithm identifies active nodes, which are nodes that (dis)appear or are linked to an edge that (dis)appears, and compares those nodes' neighborhoods with their previous time step to reassign nodes to new communities if necessary. TILES, is an online algorithm that identifies overlapping communities by iteratively recomputing a node's community membership in case of a new interaction [17]. The approach is capable of singling out community evolution events such as birth, death, merge, split, growth and contraction. OLCPM is an online, deterministic and DCD method based on clique percolation and label propagation [18]. OLCPM, unlike CPM (Clique Percolation Method) [19], works by updating

communities by looking at some predefined events resulting in improved computation times. OLCPM is able to detect overlapping communities in temporal networks. Finally, DOCET [20] incrementally updates overlapping dynamic communities after it finds an initial community structure. It can track community evolution events.

Informed CD by multi-objective optimization (2.3) The two previous categories updated partitions by looking at past communities. Informed community detection algorithms, on the other hand, calculate the communities from scratch in each time step. The algorithm tries to balance partition quality and temporal partition coherence or in other words, the current network structure and past partitions. A disadvantage of these kinds of approaches is the computational power necessary to execute the algorithm. An advantage is its temporal independence, potentially resulting in more stable outcomes. In informed community detection by multi-objective optimization, the partition at time t is detected by optimizing a certain metric, e.g. modularity, density. Two algorithms will represent this category in the evaluation. FacetNet was a pioneer in detecting communities in an unified process, in contrast with a two-step approach, where evolution events can be uncovered together with the partitioning [6]. Consequently, FacetNet is used as a benchmark approach in many papers introducing algorithms with similar capabilities. The approach finds communities based on non-negative matrix factorization and iteratively updates the network structure to balance the current partitioning fit and historical cost function. A disadvantage of the technique is that the number of communities is fixed and should be determined by the user. DYNMOGA [21], unlike FacetNet, balances the

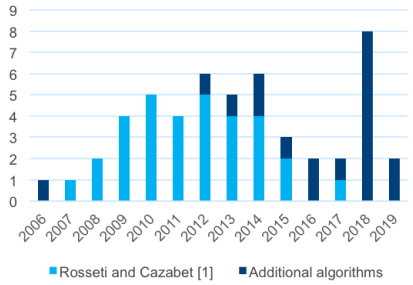

Fig. 1. Analyzed papers by year.

current partitioning fit and cost function simultaneously and, therefore, does not need a preference parameter with regard to maximizing partition quality and minimizing the historical cost or clustering drift. It optimizes a multi-objective problem and automatically determines the optimal trade-off between cluster accuracy and clustering drift. Neither FacetNet or DYNMOGA are capable of detecting overlapping communities.

Informed CD by network smoothing (2.4) ECSD proposed by [22] is a particle-and-density based evolutionary clustering method that is capable of determining the number of communities itself. The method detects the network's structure and evolutionary

events by trading off historic quality and snapshot quality, similar to the previous subcategory. The difference, however, is that ECSD finds its clusters by temporally smoothing data instead of results.

Other benchmarks Within this final category of methods introduced for comparison purposes we consider two algorithms: DEMON and iLCD. DEMON introduced in [23] is a technique that is able to hierarchically detect overlapping communities but cannot, unlike all previous methods, identify community evolution events. iLCD [24], in previous empirical comparisons, is repeatedly shown to perform worse in terms of partition quality and computation speed with regards to other tested algorithms (e.g. FacetNet). It will be interesting to evaluate or verify the relative performance of these methods.

3.2 Qualitative results

The first aspect that stands out is the larger presence of algorithms that update communities by a set of rules (2.2) not only in our final selection, but, likewise, among the more recently proposed methods, such as AFOCS, HOCTracker, OLCPM and DOCET which are also more focused on performing in dynamic social environments.

The second aspect that attracted our attention was the fact that nearly none of the analyzed algorithms focused in particular on the detection of hierarchical communities. Moreover, even though it was expected that hierarchy would be a relevant factor, it was generally not even mentioned whether an algorithm was capable of detecting hierarchical communities. On the other hand, detecting overlapping communities in social networks was oftentimes considered as necessity in current literature.

Thirdly, it is striking that all algorithms from the categories that optimize an objective function use modularity as community definition. Algorithms that do not optimize an objective function sometimes still utilize a metric as a guide to search for communities, but operate by exploiting other characteristics of the network topology, such as the frequency of node neighbors by labeling nodes, done by label propagation [23].

An overview of previously discussed characteristics for selected methods can be found in Table 1. In the last column of Table 1, time complexity per each method, as provided by its introductory study, is presented. It can be seen that the required resources needed for running Extended BGL, LabelRankT, ECSD and iLCD grow linearly with the number edges, making these algorithms the most efficient ones. Next, FacetNet's computation time grows proportional with its number of edges and communities, DYNMOGA scales log-linearly with the number of nodes and DEMON is dependent on the number of nodes and the maximum degree. Finally, TILES, AFOCS, HOCTracker and DOCET appear to be the most complex algorithms as their computation time is expected to grow quadratically with the number of nodes, which is particularly problematic for large graphs. It cannot be derived whether the complexity is closely related to the category the algorithm belongs to. Category 2.2., however, seems to be most complex.

In the context of social networks (and not only these), the knowledge of what types of community evolution events occur at which moments in time can be valuable information in order to understand what is happening with the network structure over time. Currently, the literature recognizes eight community evolution events, namely birth, death, merge, split, growth, contraction, continue and resurgence of a community, although,

Table 1. Comparison of **dynamic** methods based on observed characteristics and their time complexity (last column). "CS Type" stands for category in Cazabet and Rossetti's framework. A "-" denotes that methods do not search communities by optimizing a metric, but operate by exploiting characteristics of network topology. Notation used for time complexity: n, m, c, g - number of nodes, edges, communities and generations respectively, K - maximum degree, α - degree distribution parameter, RQ_{ttl} - expected average size of interactions processed during interaction removal phase, $|U|$ - number of nodes to be updated.

CS Type	Method	Definition	Overlap	Evolution	Hierarchy	Time Complexity
2.1	D-GT	Modularity	?	✓	?	-
	Extended BGL	Modularity	?	✓	?	O(m)
2.2	TILES	-	✓	✓	?	$O(\|n(c_u + c_v)\| + RQ_{ttl}\|U\|^2)$
	AFOCS	Density	✓	✓	?	$O(n^2)$
	HOCTracker	Density	✓	✓	✓	$O(n^2)$
	OLCPM	Modularity	✓	✓	?	-
	DOCET	-	✓	✓	?	$O(n^2)$
	LabelRankT	-	✗	✓	?	$O(m)$
2.3	FacetNet	Modularity	✗	✓	?	$O(m \cdot c)$
	DYNMOGA	Modularity	✗	✓	?	$O(gn \cdot log(n))$
2.4	ECSD	Modularity	✗	✓	?	$O(m)$
Other	Demon	-	✓	✗	✓	$O(nK^{3-\alpha})$
	iLCD	-	✓	✓	?	$O(m)$

obviously, not every method is able to detect all of them. Therefore, for each of the methods which do support community evolution tracking (see column "Evolution" in Table 1), it is worth investigating further which in particular event(s) the tracking refers to. As can be seen from Table 2, several remarks can be made along these lines.
Firstly, it is remarkable that the event resurgence cannot be detected by any of the selected algorithms, nor by any of the other algorithms that were analyzed, even though the event has been included in the literature, among others by [1]. Similarly, the event continue is rarely mentioned explicitly. It might be the case that continue is implied/detected when no event occurs and is therefore not mentioned by the authors.
Secondly, the algorithms, such as OLCPM, HOCTracker and DOCET, that were included in addition to the survey by [1] because they were more recent and possessed good features for social network community detection, can detect most of the events community evolution events. Only resurgence cannot be detected by any of the methods which we assume is due to the fact that detecting resurgence requires more than two timestamps which is not the case with methods from category 2.2, in general.
Thirdly, some algorithms, such as Extended BGL, ECSD and iLCD, only track events that are linked with the emergence of nodes and not their disappearance.

3.3 Quantitative results

In this section we present the empirical results on synthetic dataset, RDyn, and real-world dataset, DBLP, in terms of partition quality (NF1) and computation times (secs).

Synthetic graph (RDyn)

Table 2. Tracking of community evolution events by selected algorithms. "CS Type" stands for category in Cazabet and Rossetti framework. Question marks denote that the algorithm is able to detect community evolution events, but the original papers do not specify which ones explicitly.

CS Type	Method	Birth	Death	Merge	Split	Growth	Contraction	Continue	Resurgence
2.1	D-GT	✓	✓	✗	✗	✓	✓	✓	✗
	Extended BGL	✓	✗	✓	✗	✓	✗	✓	✗
2.2	TILES	✓	✓	✓	✓	✓	✓	✗	✗
	AFOCS	?	?	?	?	?	?	?	?
	HOCTracker	✓	✓	✓	✓	✓	✓	✗	✗
	OLCPM	✓	✓	✓	✓	✓	✓	✓	✗
	DOCET	✓	✓	✓	✓	✓	✓	✗	✗
	LabelRankT	✗	✓	✓	✓	✗	✗	✗	✗
2.3	FacetNet	?	?	?	?	?	?	?	?
	DYNMOGA	✓	✓	✓	✓	✓	✓	✗	✗
2.4	ECSD	✓	✗	✓	✗	✓	✗	✗	✗
Other	Demon	✗	✗	✗	✗	✗	✗	✗	✗
	iLCD	✓	✗	✓	✗	✓	✗	✗	✗

Partition quality The results of partition quality in terms of NF1 measure are provided in Table 3. The best performing algorithm is HOCTracker followed by iLCD and DEMON which only slightly differ from each other. Next, OLCPM is the second worst performer followed by Tiles who ended up having very poor results in terms of NF1. In general, the community size distribution parameter and the number of nodes do not have a trend that influences the partition quality. The impact of these variables differs algorithm to algorithm.

HOCTracker returns the highest NF1 values for $\alpha = 3$ and the lowest for $\alpha = 2.5$. However, it also exhibits much higher standard deviations associated with each group of RDyn instances, especially in comparison with iLCD and OLCPM. Note that standard deviation in Table 3 is not the standard deviation of the mean NF1 across every RDyn instance of one of the nine RDyn categories, but represents the average standard deviation of all NF1 measures within one RDyn instance. Even though the small standard deviation values in iLCD could be interpreted as method consistency (thus in its advantage), closer investigation revealed that oftentimes a lot of nodes where not assigned to a community for a specific graph resulting in NF1 scores of 0 for those communities and consequently for their graphs. If NF1 mean is 0 than the standard deviation is either close to 0 or 0. This can be seen in Figure 2. OLCPM and iLCD appear to classify algorithms quite well once the algorithm succeeds at assigning the majority of the nodes.

Scalability As can be seen from Table 4, the best performing algorithm on synthetic dataset in terms of execution times is iLCD, followed by TILES, DEMON, OLCPM and lastly HOCTracker. Remarkably, the group with $\alpha = 3$ takes the longest to execute across all sizes with the exception of OLCPM on the (1000, 3) graph where (1000, 2.5) requires the longest time. Although this observation is very notable, there is no reasonable explanation why this occurs. It can be concluded that specific characteristics

can have a significant impact on the time required to analyze a graph but we refrain from specifying a specific relation between community size distribution and execution times.

Another interesting observation is that within each size group, the graphs with relatively large differences in community sizes ($\alpha = 3.5$) often require the least time to analyze. If they are not the fastest their performance is rather similar to the fastest.

According to literature, iLCD scales with the number of edges in a graph. However, this is not reflected in the execution times. For OLCPM, it was observed that the execution time is very variable. For the $\alpha = 2.5$, the execution times for 1000 and 4000 are almost equal. For 2000 nodes it only takes 75% of the time used for the former. An anomaly can clearly be observed for OLCPM, the average execution time for the (2000, 2.5) and (2000, 3.5)-instances group takes less time than their (1000, 2.5) and (1000, 3.5) despite having double the number of nodes and approximately edges. This observation cannot be attributed to a specific aspect of the algorithm. As expected, DEMON was found to scale with the number of nodes and the average degree distribution parameter (kept constant on all instances of RDyn). At last, HOCTracker performed the worst which was not unsurprising as it scales in a quadratic way.

Real-world graph (DBLP) Three algorithms were run on the DBLP dataset: DEMON, iLCD and TILES. HOCTracker returns an OutOfMemoryException when trying to run it on the DBLP dataset, which demonstrates the unsuitability of the algorithm for large graphs. OLCPM runs itself into a loop on the DBLP dataset. We also suspect the method unsuitability for large datasets as this phenomenon did not occur on the RDyn dataset and the test in its introductory paper encompassed only small datasets ($< 10\,000$ nodes).

To analyze the performance of DEMON, iLCD and TILES each partitioning is benchmarked with the resulting partitioning of each of the other algorithms as ground truth. From the results, in Table 5, it can be observed that, on average, TILES is the worst and DEMON the best performing algorithm. Even though DEMON is the best performing algorithm, it needs significantly more computation time 3099.48 seconds on DBLP dataset as compared to TILES requiring 1436.71 seconds and iLCD which is the fastest with only 55.73 seconds. Figure 3 shows the evolution of mean NF1 scores of the three different methods for each year from 1971 to 2002. A general trend can be observed: as time progresses and more nodes and edges are introduced, the NF1 values drop significantly, however, not at the same pace for every algorithm. While TILES starts as the worst-performing algorithm in the earlier timestamps and thus smaller graphs, it ends up being the most performing one once the graph size exceeds 35 000 nodes (1991).

4 Related Work

A plethora of studies focusing purely on comparing algorithms for static community detection methods can be found in the literature [3, 8–12]. In contrast, to the best of our knowledge, there are no studies that focus purely on the empirical comparison of DCD algorithms. Instead, in the studies which introduce new DCD algorithms or new dynamic benchmark graphs, authors typically benchmark their own method with few others with the aim to showcase that their technique performs better with regard to peers.

Table 3. Mean NF1 results and the associated standard deviations of benchmarked algorithms on RDyn dataset. The highest scores per different number of nodes and alpha are underlined while the best averages are boldfaced. "CS Type" stands for category in Cazabet and Rossetti framework.

CS Type	Method	Alpha	Nodes							
			1000		2000		4000		Avg.	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
2.2	TILES	avg.	0.2002	0.2035	0.1961	0.1967	0.2087	0.1954	0.2017	0.1985
		2.5	0.1951	0.2068	0.1969	0.1994	0.2188	0.2017	0.2036	0.2026
		3	0.2033	0.2026	0.1882	0.1971	0.1953	0.1960	0.1954	0.1985
		3.5	0.2022	0.2010	0.2043	0.1934	0.2131	0.1882	0.2066	0.1941
	HOCTracker	avg.	**0.4600**	0.2596	**0.4236**	0.2397	0.3852	0.2355	0.4236	0.2402
		2.5	0.3839	0.2570	0.4095	0.2443	0.3605	0.2474	**0.4070**	0.2447
		3	0.5346	0.2536	0.4493	0.2442	0.4352	0.2272	**0.4515**	0.2440
		3.5	0.4469	0.2675	0.4096	0.2298	0.3714	0.2229	**0.4101**	0.2309
	OLCPM	avg.	0.3132	0.0243	0.3118	0.0206	0.3257	0.0192	0.3169	0.0213
		2.5	0.3006	0.0262	0.3173	0.0201	0.3274	0.0202	0.3151	0.0222
		3	0.3222	0.0253	0.3043	0.0225	0.3227	0.0190	0.3163	0.0223
		3.5	0.3171	0.0212	0.3143	0.0189	0.3270	0.0182	0.3196	0.0194
Other	DEMON	avg.	0.4022	0.3059	0.3664	0.2962	0.3687	0.2970	0.3788	0.2996
		2.5	0.3987	0.3135	0.3615	0.2974	0.3846	0.2956	0.3814	0.3021
		3	0.3952	0.3001	0.3543	0.2898	0.3396	0.2996	0.3624	0.2964
		3.5	0.4136	0.3041	0.3857	0.3021	0.3845	0.2957	0.3944	0.3006
	iLCD	avg.	0.3961	0.0170	0.3797	0.0125	**0.3968**	0.0097	0.3908	0.0130
		2.5	0.3838	0.0205	0.3826	0.0133	0.4018	0.0106	0.3894	0.0147
		3	0.3999	0.0159	0.3677	0.0131	0.3826	0.0090	0.3829	0.0126
		3.5	0.4054	0.0145	0.3908	0.0109	0.4071	0.0094	0.4012	0.0115

Table 4. Average computation time (in sec) over 100 RDyn instances per each benchmarked algorithm (the shortest boldfaced). "CS Type" stands for category in Cazabet and Rossetti framework.

CS Type	Method	Computation time (sec)			
		Alpha	Nodes		
			1000	2000	4000
2.2	TILES	2.5	3.43	5.45	7.47
		3	3.53	6.05	8.42
		3.5	3.45	5.48	7.73
	HOCTracker	2.5	46.12	77.77	152.00
		3	49.61	94.22	214.44
		3.5	43.96	78.62	166.25
	OLCPM	2.5	39.92	30.69	41.74
		3	37.48	69.27	56.40
		3.5	32.79	29.37	45.55
other	DEMON	2.5	11.34	18.68	30.42
		3	11.77	20.8	35.02
		3.5	10.38	16.81	28.98
	iLCD	2.5	**1.94**	**2.35**	**2.74**
		3	**2**	**2.53**	**3.08**
		3.5	**1.98**	**2.37**	**2.81**

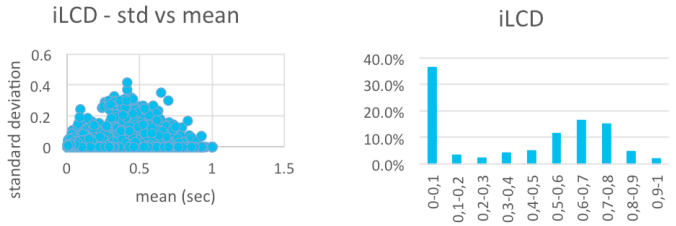

Fig. 2. NF1 mean vs. standard deviation for iCLD on synthetic data.

The situation is slightly better with respect to benchmark graphs, where a vast body of literature is available. Although the most prominently used benchmark graphs Girvan-Newman (GN) [4] and Lancichinetti-Fortunato-Radicchi (LFR) [2] are not suited for temporal community discovery, to this end, their extensions in [6] and [5] respectively, were proposed. Next, RDyn, a framework for generating dynamic networks along with time-dependent ground-truth partitions with tunable qualities, was introduced in [1].

To gain better insight in how the mentioned sporadic comparisons of DCD algorithms and/or dynamic benchmark graphs have been performed, in the context of this literature review, we considered a selection of 51 papers on DCD methods. The first remarkable finding is that 12 out of 51 papers did not include a single comparison with peer algorithms [28, 33, 35, 40, 41, 44, 45, 48–50, 52, 56], while 39 did. A closer investigation of these 39 papers revealed 57 algorithms out of 82 were only benchmarked once. On the other hand, the most frequently referenced algorithm is FacetNet [6]. The second interesting finding is that authors seem to use various datasets for comparison, as they often include synthetic graphs and/or real-world graphs. In the assessed papers, 1 made use of only synthetic graphs [32], 32 used only real graphs [13, 14, 16, 22, 27, 28, 30, 31, 33, 34, 36–41, 44, 45, 48–52, 54–57, 59–63] and 17 used both [6, 15, 17, 18, 20, 21, 23, 24, 29, 42, 43, 46, 47, 53, 58, 64, 65]. In the 49 papers that used real graphs 47 different real graphs were introduced. A little over half of the datasets are only used once. The most popular are graphs extracted from the DBLP database. These occur as benchmark graphs in 19 of 51 papers. Hence, similarly to the use of methods, the use of datasets is also fairly heterogeneous which contributes to the difficulty of assessing the relative performance of techniques.

Due to the fact that the overlap in comparison is limited, it is hard to make any deductions with regard to the relative performance of the algorithms. Moreover, as mentioned before, the setup and results of these comparisons might contain an unconscious bias towards the proposed algorithm. This shows the relevance of this study.

5 Conclusion

Dynamic community detection has numerous applications in different fields and as such is extensively studied in the current literature. Nevertheless, a systematic and unbiased comparison of these methods is still missing. Therefore, in this paper we made steps towards scrutinizing algorithms and performing fairly both qualitative and quantitative

Table 5. The mean NF1 results and the associated standard deviations when running DEMON, iLCD and TILES (rows) and using them as ground truth (columns) on DBLP dataset.

Method	Ground truth						Avg
	TILES		DEMON		iLCD		
	Mean	Std.	Mean	Std.	Mean	Std.	
TILES	1.0000	0.0000	0.5840	0.3211	0.5941	0.3182	0.5891
DEMON	0.5369	0.2612	1.0000	0.0000	0.7612	0.2817	**0.6490**
iLCD	0.5117	0.2685	0.6619	0.3116	1.0000	0.0000	0.5868

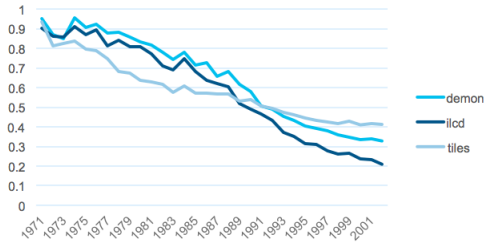

Fig. 3. Mean NF1 results for 1971-2002.

comparisons on synthetic as well as real-life evolving graphs. The qualitative analysis included an overall set of characteristics relevant for (social) community detection such as community definition used, the ability to track community life-cycle events, overlapping and hierarchical communities, and the time complexity. For the empirical analysis, several limiting factors such as unavailable/poorly documented source code and inability to run methods on large graphs led to a narrower set of compared methods. Nevertheless, 900 synthetic, evolving graphs of various sizes and community size distributions and the most frequently used real-world DBLP dataset were used for a thorough analysis.

Undoubtedly, the field of community detection techniques that act on evolving graphs is characterized by its inherent heterogeneity in all its aspects. As such, there is no single best performing community technique, but rather, the choice and the performance depends on the objective and dataset characteristics. For future work, we envision an even more extensive empirical evaluation.

References

1. G. Rossetti and R. Cazabet, “Community Discovery in Dynamic Networks: A Survey,” ACM Com-put. Surv, vol. 51, 2018.
2. A. Lancichinetti, S. Fortunato, and F. Radicchi, “Benchmark graphs for testing community detection algorithms,” Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., 2008.
3. A. Lancichinetti and S. Fortunato, “Community detection algorithms: A comparative analysis,” Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., 2009.
4. M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., 2004.
5. D. Greene, D. Doyle, and P. Cunninngham, “Tracking the evolution of communities in dynamic social networks,” Soc. Networks, pp. 1-18, 2011.

6. Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Analyzing communities and their evolutions in dynamic social networks," ACM Trans. Knowl. Discov. Data, vol. 3, no. 2, pp. 1-31, 2009.
7. G. Rossetti, "RDyn: Graph benchmark handling community dynamics," J. Complex Networks, 2017.
8. L. Danon, A. Dìaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," J. Stat. Mech. Theory Exp., no. 9, pp. 219-228, 2005.
9. S. Harenberg et al., "Community detection in large-scale networks: A survey and empirical evaluation," Wiley Interdiscip. Rev. Comput. Stat., vol. 6, no. 6, pp. 426-439, 2014.
10. Z. Yang, R. Algesheimer, and C. J. Tessone, "A Comparative Analysis of Community Detection Algorithms on Artificial Networks," Sci. Rep., vol. 6, no. 1, p. 30750, Nov. 2016.
11. P. Wagenseller, F. Wang, and W. Wu, "Size Matters: A Comparative Analysis of Community Detection Algorithms," IEEE Transactions on Computational Social Systems, 2018.
12. Z. Zhao, S. Zheng, C. Li, J. Sun, L. Chang, and F. Chiclana, "A comparative study on community detection methods in complex networks," J. Intell. Fuzzy Syst., 2018.
13. H. Alvari, A. Hajibagheri, and G. Sukthankar, "Community detection in dynamic social networks: A game-theoretic approach," ASONAM 2014 - Proc. 2014 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Min., pp. 101-107, 2014.
14. J. Shang et al., "A Real-Time Detecting Algorithm for Tracking Community Structure of Dynamic Networks," vol. 12, 2014.
15. N. P. Nguyen, T. N. Dinh, S. Tokala, and M. T. Thai, "Overlapping communities in dynamic networks: their detection and mobile applications," Mobicom, pp. 85-96, 2011.
16. S. Y. Bhat and M. Abulaish, "HOCTracker: Tracking the Evolution of Hierarchical and Overlapping Communities in Dynamic Social Networks," IEEE Trans. Knowl. Data Eng., vol. 27, no. 4, pp. 1013-1019, 2015.
17. G. Rossetti, L. Pappalardo, D. Pedreschi, and F. Giannotti, "Tiles: an online algorithm for community discovery in dynamic social networks," Mach. Learn., vol. 106, no. 8, pp. 1213-1241, 2017.
18. S. Boudebza, R. Cazabet, F. Azouaou, and O. Nouali, "OLCPM: An online framework for detecting overlapping communities in dynamic social networks," Comput. Commun., vol. 123, pp. 36-51, 2018.
19. P. Gergely, D. Imre, F. Illés, and V. Tamás, "Uncovering the overlapping community structure of complex networks in nature and society," Nature, vol. 435, no. m, pp. 814-818, 2005.
20. Z. Wang, Z. Li, G. Yuan, Y. Sun, X. Rui, and X. Xiang, "Tracking the evolution of overlapping communities in dynamic social networks," Knowledge-Based Syst., vol. 157, no. December 2017, pp. 81-97, 2018.
21. F. Folino, C. Pizzuti, and V. Pietro Bucci, "A Multiobjective and Evolutionary Community Detection Method for Dynamic Networks."
22. M.-S. Kim and J. Han, "A particle-and-density based evolutionary clustering method for dynamic networks," Proc. VLDB Endow., vol. 2, no. 1, pp. 622-633, 2009.
23. M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "DEMON: a Local-First Discovery Method for Overlapping Communities," pp. 615-623, 2012.
24. R. Cazabet, F. Amblard, and C. Hanachi, "Detection of overlapping communities in dynamical social networks," Proc. - Soc. 2010 2nd IEEE Int. Conf. Soc. Comput. PASSAT 2010 2nd IEEE Int. Conf. Privacy, Secur. Risk Trust, no. August, pp. 309-314, 2010.
25. E. Demaine and M. Hajiaghayi, "DBLP Graphs (BigDND: Dynamic Network Data)," 2019. [Online]. Available: http://projects.csail.mit.edu/dnd/DBLP/.
26. M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "Uncovering Hierarchical and Overlapping Communities with a Local-First Approach," ACM Trans. Knowl. Discov. Data, vol. 9, no. 1, pp. 1-27, 2014.

27. T. Aynaud and J.-L. Guillaume, “Static community detection algorithms for evolving networks,”, Ad Hoc Wireless Networks, pp. 513-519, 2010.
28. Shweta Bansal, Sanjukta Bhowmick, and Prashant Paymal, “Fast Community Detection for Dynamic Complex Networks,” Commun. Comput. Inf. Sci., vol. 116, pp. 196-207, 2011.
29. R. Görke, P. Maillard, C. Staudt, and D. Wagner, “Modularity-Driven Clustering of Dynamic Graphs. Lecture Notes in Computer Science,” Sea, vol. 6049, no. Chapter 37, pp. 436-448, 2010.
30. K. Miller and T. Eliassi-Rad, “Continuous Time Group Discovery in Dynamic Graphs,” Networks Learn. with Graphs, pp. 1-7, 2009.
31. M. K. Agarwal, K. Ramamritham, and M. Bhide, “Real Time Discovery of Dense Clusters in Highly Dynamic Graphs: Identifying Real World Events in Highly Dynamic Environments,” vol. 5, no. 10, pp. 980-991, 2012.
32. R. Cazabet and F. Amblard, “Simulate to detect: A multi-agent system for community detection,” Proc. - 2011 IEEE/WIC/ACM Int. Conf. Intell. Agent Technol. IAT 2011, vol. 2, no. September, pp. 402-408, 2011.
33. D. Duan, Y. Li, R. Li, and Z. Lu, “Incremental K-clique clustering in dynamic social networks,” Artif. Intell. Rev., vol. 38, no. 2, pp. 129-147, 2012.
34. T. Falkowski, A. Barth, and M. Spiliopoulou, “DENGRAPH: A density-based community detection algorithm,” in WI ’07 Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, 2007, pp. 112-115.
35. R. Görke, T. Hartmann, and D. Wagner, “Dynamic graph clustering using minimum-cut trees,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5664 LNCS, no. 2, pp. 339-350, 2009.
36. P. Lee, L. V. S. Lakshmanan, and E. E. Milios, “Event Evolution Tracking from Streaming Social Posts,” 2013.
37. J. Huang, “CUT: Community Update and Tracking in Dynamic Social,” no. 1.
38. N. P. Nguyen, T. N. Dinh, Y. Xuan, and M. T. Thai, “Adaptive algorithms for detecting community structure in dynamic social networks,” in 2011 Proceedings IEEE INFOCOM, 2011, pp. 2282-2290.
39. J. Xie, M. Chen, and B. K. Szymanski, “LabelRankT: Incremental Community Detection in Dynamic Networks via Label Propagation,” 2013.
40. A. Zakrzewska and D. A. Bader, “A Dynamic Algorithm for Local Community Detection in Graphs,” Proc. 2015 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Min. 2015 - ASONAM ’15, no. 5, pp. 559-564, 2015.
41. H. Crane and W. Dempsey, “Community detection for interaction networks,” pp. 1-29, 2015.
42. M. G. Gong, L. J. Zhang, J. J. Ma, and L. C. Jiao, “Community detection in dynamic social networks based on multiobjective immune algorithm,” J. Comput. Sci. Technol., vol. 27, no. 3, pp. 455-467, 2012.
43. R. Görke, P. Maillard, A. Schumm, C. Staudt, and D. Wagner, “Dynamic graph clustering combining modularity and smoothness,” J. Exp. Algorithmics, vol. 18, no. April, pp. 1.1-1.29, 2013.
44. V. Kawadia and S. Sreenivasan, “Sequential detection of temporal communities by estrangement confinement,” Sci. Rep., vol. 2, 2012.
45. Y. Sun, J. Tang, J. Han, M. Gupta, and B. Zhao, “Community evolution detection in dynamic heterogeneous information networks,” Proc. Eighth Work. Min. Learn. with Graphs - MLG ’10, pp. 137-146, 2010.
46. L. Tang, H. Liu, J. Zhang, and Z. Nazeri, “Community evolution in Dynamic Multi-Mode Networks,” Tetrahedron Lett., vol. 45, no. 9, pp. 1903-1906, 2004.
47. T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, “Detecting communities and their evolutions in dynamic social networksâĂŤa Bayesian approach,” Mach. Learn., vol. 82, no. 2, pp. 157-189, Feb. 2011.

48. D. Zhou, I. Councill, H. Zha, and C. L. Giles, "Discovering temporal communities from social network documents," Proc. - IEEE Int. Conf. Data Mining, ICDM, pp. 745-750, 2007.
49. C. Guo, J. Wang, and Z. Zhang, "Evolutionary community structure discovery in dynamic weighted networks," Phys. A Stat. Mech. its Appl., vol. 413, pp. 565-576, 2014.
50. K. S. Xu and A. O. Hero, "Dynamic stochastic blockmodels: Statistical models for time-evolving networks," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7812 LNCS, pp. 201-210, 2013.
51. J. Li, L. Huang, T. Bai, Z. Wang, and H. Chen, "CDBIA: A dynamic community detection method based on incremental analysis," in 2012 International Conference on Systems and Informatics (ICSAI2012), 2012, pp. 2224-2228.
52. D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06, 2006, p. 554.
53. Y. Hu, B. Yang, and C. Lv, "A local dynamic method for tracking communities and their evolution in dynamic networks," Knowledge-Based Syst., vol. 110, pp. 176-190, 2016.
54. N. Ilhan and I. G. Oguducu, "Community Event Prediction in Dynamic Social Networks," in 2013 12th International Conference on Machine Learning and Applications, 2013, pp. 191-196.
55. A. P. Appel, R. L. F. Cunha, C. C. Aggarwal, and M. M. Terakado, "Temporally evolving community detection and prediction in content-centric networks," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 11052 LNAI, pp. 3-18, 2019.
56. E. G. Tajeuna, M. Bouguessa, and S. Wang, "Modeling and Predicting Community Structure Changes in Time Evolving Social Networks," IEEE Trans. Knowl. Data Eng., vol. PP, no. c, p. 1, 2018.
57. Z. Zhao, C. Li, X. Zhang, F. Chiclana, and E. H. Viedma, "An incremental method to detect communities in dynamic evolving social networks," Knowledge-Based Syst., vol. 163, pp. 404-415, 2019.
58. P. Jiao, W. Wang, and D. Jin, "Constrained common cluster based model for community detection in temporal and multiplex networks," Neurocomputing, vol. 275, pp. 768-780, 2018.
59. H. S. Cheraghchi and A. Zakerolhosseini, "Toward a novel art inspired incremental community mining algorithm in dynamic social network," Appl. Intell., vol. 46, no. 2, pp. 409-426, 2017.
60. H. Sun et al., "IncOrder: Incremental density-based community detection in dynamic networks," Knowledge-Based Syst., vol. 72, pp. 1-12, 2014.
61. A. Said, R. A. Abbasi, O. Maqbool, A. Daud, and N. R. Aljohani, "CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks," Appl. Soft Comput. J., vol. 63, pp. 59-70, 2018.
62. Z. Li, J. Liu, and K. Wu, "A multiobjective evolutionary algorithm based on structural and attribute similarities for community detection in attributed networks," IEEE Trans. Cybern., vol. 48, no. 7, pp. 1963-1976, 2018.
63. M. Asadi and F. Ghaderi, "Incremental community detection in social networks using label propagation method," in Conference of Open Innovations Association (FRUCT), 2018, pp. 13-16 Nov. 2018.
64. Y. Li, K. He, K. Kloster, D. Bindel, and J. Hopcroft, "Local Spectral Clustering for Overlapping Community Detection," ACM Trans. Knowl. Discov. Data, vol. 12, no. 2, pp. 1-27, 2018.
65. C. Zhang, Y. Zhang, and B. Wu, "A parallel community detection algorithm based on incremental clustering in dynamic network," Proc. 2018 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Mining, ASONAM 2018, pp. 946-953, 2018.