Journal of Complex Networks (2017) **0000**, 1–20
doi: 10.1093/comnet/cnx016

RDyn: graph benchmark handling community dynamics

Giulio Rossetti[†]
University of Pisa & ISTI-CNR, Italy
[†]Corresponding author. Email: giulio.rossetti@di.unipi.it

Edited by: Guido Caldarelli

[Received on 6 July 2016; editorial decision on 23 May 2017; accepted on 29 May 2017]

Graph models provide an understanding of the dynamics of network formation and evolution; as a direct consequence, synthesizing graphs having controlled topology and planted partitions has been often identified as a strategy to describe benchmarks able to assess the performances of community discovery algorithm. However, one relevant aspect of real-world networks has been ignored by benchmarks proposed so far: community dynamics. As time goes by network communities rise, fall and may interact with each other generating merges and splits. Indeed, during the last decade dynamic community discovery has become a very active research field: in order to provide a coherent environment to test novel algorithms aimed at identifying mutable network partitions we introduce RDyn, an approach able to generates dynamic networks along with time-dependent ground-truth partitions having tunable quality.

Keywords: graph models; dynamic networks; evolving communities.

1. Introduction

Our world is permeated by connected systems that can be naturally represented as networks: social relationships [2, 5], technology networks [1], chemical networks [24], the World Wide Web [33] are all examples of complex realities in which entities are related one to another following clear semantics. The analysis of such complex systems nowadays represents a hot topic studied from multiple perspectives: computer scientists, physicists, mathematicians, biologists, researchers from countless fields are used to model their worlds with the tools offered by graph theory in order to impose a structure to otherwise chaotic data. In this challenging scenario, identifying shared methodologies to define and build effective controlled environments for testing purposes is mandatory: for this reason, benchmarks have been proposed to evaluate and compare the performances of algorithmic solutions designed to address countless network problems. Indeed, real-world networks, as described by heterogeneous datasets, have repeatedly shown to follow some peculiar characteristics: network generators able to reproduce such traits, producing tunable network topologies, are the basic bricks on which controlled experiments pose their grounds.

Nowadays, one of the most intriguing topics in complex network analysis is community discovery (CD) [15], that is the problem of automatically identify meaningful partitions of a given graph by grouping together entities that are strictly related within each other than with the rest of the network. Indeed, community discovery is an ill-posed problem since a formal shared definition of communities does not exist: however, its countless applications made it one of the most active publication themes in this multidisciplinary community. In order to evaluate the effectiveness of a CD approach, it is usual to compare the identified partition with a ground truth one. Since real data rarely come with annotated ground truth partitions in the last decades several graph generators [16, 18, 20, 26, 39] were designed in

order to produce benchmarks having topological characteristics similar to the real world ones as well as planted ground truth communities.

However, a major assumption was made so far by almost all the proposed generators: the networks modelled are static and the communities do not change as time goes by. Recently, dynamic CD [9] (henceforth, DCD) has emerged as an extension of the classical CD problem. As the phenomena they model, real networks evolve through time: as in a social network a user can interact multiple times with a friend as well as destroy a friendship tie, in a dynamic graph a node can establish multiple interactions with the same node or delete a pre-existent edge. Nodes, as well as edges, may appear and disappear and the perturbations such events cause to the network topology usually reshape the community structures it embeds. DCD approaches track such changes over time [10, 36]: classic benchmarks that assume both the network topology and planted community static are arguably, at least in this scenario, the best solution while designing controlled environment testing.

In this work, we introduce RDyn, a novel benchmark specifically tailored to simulate a completely dynamic scenario. RDyn generates dynamic graphs respecting well-known real-world network properties, proposing an approach able to build tunable quality evolving ground truths (i.e. allowing both community merge and split events). RDyn goal is to provide a benchmark for DCD algorithms: to do so it regulates both community merge/split events and generates stable community states (i.e. the ground truth) by continuously checking partition quality as the network topology evolves through time.

The article is organized as follows: in Section 2 are discussed the related works; in Section 3 RDyn is introduced; in Section 4 is described a novel benchmark that uses RDyn as network generator and a novel score, NF1[37], as quality function. In Section 5, RDyn is evaluated against state of art CD and DCD algorithms as well as against LFR [27] one of the most famous benchmarks for static graphs: moreover, in this section both the RDyn planted communities characteristics and execution times are discussed. Finally, Section 6 concludes the article.

2. Related works

Identify a reliable way to evaluate partition quality is one of the major issues to address while approaching CD. Due to the absence of a shared formal definition of what is a community and which characteristics it needs to possess, each paper follows its own strategy to compare the results obtained by its algorithm with the ones produced by state of art methods. In this section, we will review some of the most popular strategies used to define testing environments aimed at evaluating community partitions. In particular, we will shed some lights on the most famous synthetic network generators used to set up controlled experiments as well as on classic network models designed to embed community structures. Moreover, we will also briefly introduce the concept of *dynamic community* and discuss some general findings on such subject.

Network models embedding community structures. The general aim of network modelling is to capture some essential properties lying behind real-world phenomena and replicate them while generating synthetic data, all imposing only a few simple constraints. In order to do so, several models were defined to generate networks embedding some of their peculiar characteristics: small-world [41], scale-free degree distribution [4], high clustering coefficients and community structure [29].

In [44], the authors extend the preferential attachment model by introducing global random attachment for community selection. The model works in two phases: (i) each new node selects a community randomly from the existing ones in the network then (ii) it connects to other nodes in such community following the Barabasi–Albert model. This approach is designed to build graphs having power law degree distribution

that, differently from the original model, embed community structures. Following a similar rationale, in [43] and [49] node-community assignations are achieved, during the first phase, by applying preferential attachment, thus biasing the selection towards big communities. This strategy enables the authors to guarantee power law distributions for both community size and node degree. A similar approach is applied even in [23] and [35] where the authors were able to generate communities having a hierarchical structure, internal power law degree distribution, and high clustering coefficient. Conversely, from previous works, [46] and [47] focus their attentions on other interesting characteristics: randomness and overlap. In [46], the authors found that introducing randomness in an Erdös–Rény graph it is possible to obtain clustered networks while losing the scale-free property. In [47] instead are introduced in the generation process the concept of introversion–extroversion, homophily and tie strength. Such model suggests the existence of, possibly overlapping, communities having different sizes. Following a completely different approach in [25] is proposed a model that generates a network with moderate size communities using local scale rules borrowed from sociology. The approach exploits triadic closure as well as focal closure in order to assure acquaintances at a local scale and random walks to capture cyclic closure and thus strong ties.

Synthetic benchmarks. Complex network modelling studies generated a new field of research: synthetic network generators. Generators allow scientists to evaluate their algorithms on synthetic data whose characteristics resemble the ones that can be observed in real-world networks. The main rationale behind the adoption of network generators as benchmarks while analyzing the performances of a CD algorithms lies in the ability to produce datasets that allow *controlled environment testing*. Benchmarks allow to tune network characteristics, such as the network size and density: this flexibility enables an extensive algorithm evaluation on networks having different characteristics but generated to follow a similar topology. This enables, for instance, to check an algorithm against:

- *Stability:* the performance of a CD approach can be evaluated on a high number of network instances having similar properties in order to provide an estimate of the algorithmic stability;
- *Scalability:* synthetic graphs can be used to test the actual scalability of an algorithm while increasing the network size.

Moreover, benchmarks provide ground-truth partitions explicitly embedded in the generated graph: such partitions can be used to evaluate the fitness of the one produced by the tested algorithm.

Among the benchmarks, the most used are the ones proposed in [18] and [27]. The Girvan–Newman benchmark, GN, is built upon the so-called *planted l-partition* model (described in: [8, 11, 17]): this generator takes as input a given ground truth partition and two parameters identifying respectively the probabilities of intra and inter-clusters links. Moreover, the ground truth partitions are generated as equally sized Erdös–Reńy random graphs [14]. The aim of GM benchmark is to assess the degree of adherence the communities identified by a given algorithm have w.r.t. the planted ground truth. Typically the analysis is performed varying the intra/inter-cluster probabilities so to characterize the topology of hidden partitions a specific algorithm is able to identify. *Planted l-partition* models (also known as *ad-hoc* models) were proposed to produce graphs resembling different real-world characteristics: among them we recall ad-hoc models that generate overlapping community structures [39] as well as weighted [16] and bipartite [20] graphs. The main drawbacks introduced by the GN benchmark (as highlighted in [27]) are twofold:

- all nodes of the network have essentially the same degree (due to the use of ER graphs);
- the planted communities have fixed and equal sizes (even though a variant of the original approach handling communities of different sizes was introduced in [13]).

Such peculiarities make the generated benchmark graphs unrealistic proxies for real networks with community structure. Indeed, real networks are characterized by heterogeneous distributions of node degree, whose tails often decay as power laws. To cope with the limitation of the GN benchmark, and to fill the gap among Erdös–Rény graphs and real ones, in [27] was introduced the LFR benchmark[1]. The networks generated by this model have both node degrees and community sizes following a power law distribution. Similar to the *planted l-partition* model, vertices share a certain fraction of their links with other vertices in their cluster and the remaining links with random vertices in other parts of the graph. Moreover, LFR allows the analyst to decide the average cluster density and size of the generated graph. This benchmark has been generalized to weighted and directed graphs, as well as to generate overlapping clusters, [26].

However, both the GN and the LFR benchmarks are designed to evaluate static graph partitions and do not natively support the generation/analysis of dynamic graphs. In order to cope with this limitation, few papers have proposed alternative benchmarks as well as extension. In [30] was proposed a variant of the GN model to evaluate the FaceNet framework: there, the authors introduce network dynamics by generating different graphs for each time steps. In particular, in order to interpolate from a snapshot to the subsequent one, the proposed model (i) randomly selects a fixed number of nodes from each community (ii) removes them and (iii) allocates them to different communities. Finally, edges are added randomly with a higher probability for within-community edges and a lower probability for between-community edges. In [19] is proposed a set of benchmarks based on LFR: at each step, starting from a static synthetic graph with ground-truth communities, 20% of the node memberships were randomly permuted in order to mimic node-community migrations. Finally, in [32] a network generation model based on evolution dynamics is introduced.

Dynamic Communities. DCD is a relatively novel task in complex network analysis [3, 9], its goal being identify and track trough time clusters of highly connected nodes in a dynamic network. In a preliminary survey [22], two high-level categories of online DCD algorithms are identified depending on how the community evolution is handled:

1. *Temporal smoothness* approaches run the CD process from scratch on each graph evolution step (network snapshot or interaction);
2. *Dynamic updates* approaches incrementally update the communities as time goes by looking both at their previous states and at novel network perturbations.

As in static CD, a formal and shared definition of dynamic communities is missing: each algorithm approaches the problem by proposing solutions that search for different topological characteristics. Since there are countless ways to define what a dynamic community should look like most of the literature on the subject focus on the proposal of approaches able to track communities elementary actions. With this aim, several works converged on a set of elementary events that regulate community life-cycle [10, 21, 34, 36]: birth, death, growth, contraction, merge and split. Moreover, a seventh operation, 'Continue', is sometimes added to these ones and in [9], an eighth operation was proposed (Resurgence, e.g. the reappearance of a previously vanished community). Indeed, not all such operations are necessarily handled by all DCD algorithms, however, few of them (birth, death, merge and split) are considered ubiquitous and thus needed to be taken into account while designing a benchmark for such task.

[1] Code available at: https://sites.google.com/site/santofortunato/inthepress2

TABLE 1 RDYN *parameters. Distributions of node degree, community size and interaction decay are additional parameters of the model not reported in the table*

Parameter	Description
$\|V\|$	Number of nodes
κ	Community quality threshold
p_{in}	Node intra community edge probability
ν	Renewal after decaying

3. RDyn: dynamic graph generators

Dynamic networks can be used to model a wide range of real-life phenomena, however being able to access dynamic datasets having ground-truth communities it is not trivial. To overcome this issue we designed RDYN[2] a flexible network generator able to simulate not only interaction dynamics but also community ones. The proposed approach is designed to allow its user to deeply customize the generated topology and related dynamics: in order to enforce the approach flexibility several controlling variable, listed in Table 1, are exposed. In the following of the article, when we will refer to a dynamic graph (or, equivalently, to a dynamic network) we will adopt the following notation:

DEFINITION 3.1 (GRAPH) $G = (V, E)$ is a dynamic graph where V represent the node set and E the edge, or equivalently interaction, set. An edge (interaction) $e \in E$ is univocally identified by a tuple:

$$e = (u, v, t, \tau), \tag{3.1}$$

where $u, v \in V$ are nodes, $t \in \mathbb{N}$ is the timestamp of e appearance and $\tau \in \mathbb{N}$ is e time to leave, that is the number of iterations the edge will remain active before vanishing.

We design our approach to generate dynamic graphs respecting to the following characteristics: (i) power law node degree distribution; (ii) power law community size distribution; (iii) tunable community quality (i.e. minimum conductance, high modularity, high density…); (iv) edge appearance/vanishing handling; (v) user defined distribution for interaction decay and (vi) communities merge/split dynamics.

In order to satisfy such desiderata, RDYN (whose pseudocode is reported in Algorithm 1) has been designed as an *iterative process* composed by three main components: (i) distribution configuration (Algorithm 1, lines 1–2), (ii) dynamic network generation and (Algorithm 1, lines 4–7) and (iii) community dynamic generation (Algorithm 1, lines 8–11).

We define RDYN an *iterative* process since the topologies it generates are the results of subsequent choices made by the nodes within it: more specifically, during every *iteration* the nodes in V are enabled to perform a specified set of actions (i.e. create/destroy edges—all subject to specific rules). Moreover, once completed each iteration the status of the resulting communities is evaluated, returned if considered stable, and community dynamics are planted.

[2] Python code available at: https://github.com/GiulioRossetti/RDyn

In the following, we will discuss each RDyn component in order to highlight how they concur to the definition of the whole model.

3.1 *Planted distributions*

As a first step RDyn assigns each node in $u \in V$ to an initial community. To do so three constraints need to be satisfied: (i) community size distribution, (ii) node degree distribution and (iii) expected ratio of intra/inter-community edge of community nodes. In our model, we assume community size distribution—as well as degree distribution—to follow (potentially truncated) power laws, for example to comply with the equation $f(x) = x^{-\alpha}$, having user-defined exponents and means.The minimum node degree as well as community size, given a specific α value can be computed as:

$$x_{min} = \frac{x}{2^{\frac{1}{\alpha-1}}}. \tag{3.2}$$

Such value, x_{min}, is needed in order to guarantee a finite area under the distribution curve: indeed, without such constraint the area approaches to $+\infty$ while x approaches 0. Moreover, we impose a maximum node degree of $|V| - 1$ and that the sum of community sizes equal to $|V|$.

Once fixed the exponent of node degree and community size distributions, both of them are generated: subsequently, RDyn walks through the nodes in V by decreasing degree and associates each one of them to a community C iff the following constraints are satisfied:

(i) C has at least an unassigned node slot and

(ii) C has a size s able to ensure that at least p_{in} of the degree of the current node u can be used to generate intra-community interactions

where p_{in} is a user-defined threshold introduced to impose an upper bound to the ratio of inter and intra community edges for each node. It ranges in $[0, 1]$; for $p_{in} = 1$ the resulting network will be composed of several disconnected components since each node will be connected exclusively with peers belonging to the same community; conversely for $p_{in} \to 0$ the resulting topology will interpolate toward a complete mixing scenario in which the planted communities are as dense as random connected nodes partitions. If a given node u is not directly assigned to any community (due to constraint (ii)), a community C which still has available slots and that reduces the over quota percentage of u expected inter-community interactions is selected. This greedy solution ensures that the initial distributions remain as stable as possible and that the average community cut ratio approaches p_{in}.

3.2 *Network dynamics*

Once produced an initial stable assignment of nodes to communities RDyn can start generating the edge stream that describes the desired dynamic network (Algorithm 1, lines 4–7). To support the edge streaming creation two sub-problems needs to be tackled: (i) how to handle intra/inter-community interaction generation and (ii) how to model interaction decay.

3.2.1 *Intra/inter-community edges.* This first issue can be easily broken into two subtasks, one addressing intra-community edges, the other inter-community ones. Both the subtasks concur to the same purposes: (i) guarantee that the real degree distribution approaches the planted one and (ii) assure that at

Algorithm 1 RDYN

Require: Parameters as defined in Table 1

```
1. Instantiate distributions
2. Assign nodes to communities
3. for each iteration do
4.     for each node u ∈ shuffle(V) do
5.         remove expired interactions involving u          ▷ subject to ν
6.         generate a new interaction for u                 ▷ subject to p_in
7.     end for
8.     if stable community status then                      ▷ subject to κ
9.         output communities
10.        generate merge/split actions
11.    end if
12. end for
```

any moment, and for any node, the real intra-community edge ratio $\widehat{p_{in}}$ is as close as possible to the user defined p_{in}. We designed RDYN as an iterative process and model nodes as *agents* that perform (at least) an action during each iteration.

In particular, during each iteration each node $u \in V$ will establish an interaction (iff it has not still reached its expected degree) towards a node of the same community - with probability p_{in}—or towards a node external to the community—with probability 1-p_{in}.

Intra-community interactions. If a node $u \in V$ is selected to generate a new interaction within its community C two scenarios may arise:

- First intra-community interaction: the endpoint of interaction, v, is extracted among the nodes in C having not yet reached their expected degree using a preferential attachment strategy (e.g. giving priority to the ones having higher expected degree);
- u already connected to other nodes in C: the endpoint v is chosen among the two-hop neighbourhoods of v (i.e. the nodes directly connected to u's neighbours) using a preferential attachment. If all the nodes in such set have reached their expected degree v is chosen among the other community nodes that have not saturated their degree capability (still sampling trough preferential attachment).

The former scenario handles the creation of the first intra-community interactions, the latter enforces the triadic closure—whenever possible—while still giving priority to preferential attachment to facilitate the rising of power-law degree distribution.

Inter-community interactions. Coherently with the intra-community connection strategy, inter-community bridges are established employing a preferential attachment sampling. In particular the procedure follows two steps: (i) a community C is selected as target community among the ones whose nodes have still not reached their maximum inter-community degree applying a preferential attachment sampling weighted on the community sizes and (ii) a node v among the candidate nodes of C is chosen via a preferential attachment. This approach guarantees a coherent node degree distribution and uses the size as a weight for the *attraction* power of a community: the bigger the community the more likely nodes are driven to connect with it.

3.2.2 *Interactions decay.* As described so far RDyn handle only network growth: in order to produce more realistic network dynamics, it is thus necessarily to embed in it an interaction decay strategy. To simulate interaction vanishing, every time a new edge (u, v) is generated its time-to-live (from now on τ) is extracted from a decay distribution then, as the first step of every new iteration, nodes are checked for expired interactions and, if found they are removed. Moreover, in order to assure topology stability each vanishing edge is renewed with probability ν if intra-community, $1 - \nu$ if inter-community. From every dynamic phenomenon it is possible to extract a specific interaction decay signature: indeed, several works analyzing real data identified particular decay distributions ranging from exponential [40] to long-tailed [48] ones. We designed our model to be as general as possible: to do so RDyn let the user define the desired decay function (as in [42] where PED a user-defined probabilistic edge decay model is introduced). In the experimental section we instantiate RDyn to simulate an exponential interaction decay (which has been chosen as default setting), $f(x) = e^{-\lambda x}$, assuming $\lambda = 1$.

3.3 *Community dynamics*

Once described the rules that regulate network dynamics it is mandatory to address the main goal of the introduced benchmark: propose a way to provide time-aware ground-truth communities allowed to appear, disappear, merge and split according to the arrival and departure of edges. In order to guarantee the embedding of meaningful communities in the generated graph, we decided to check their significance at each iteration through the adoption of a quality measure. With this aim, we adopted *conductance* as a way to continuously assess partition stability and identify community shifts.

Definition 3.2 (Conductance) In graph theory the conductance score [7] is used to measure how 'well-knit' a graph is: it controls how fast a random walk on a graph G converges to a uniform distribution. In CD, this measure is often used to provide a proxy to assess a partition quality [45]. Given a set of communities $\mathscr{C}$=$\{C_0, \ldots, C_i, \ldots, C_n\}$ conductance is defined as:

$$K(C_i) = \frac{\sum_{u \in C_i, v \in \overline{C_i}} a_{u,v}}{min(a(C_i), a(\overline{C_i})}, \tag{3.3}$$

where $\overline{C_i}$ identifies the nodes outside C_i and $a_{u,v}$ are the entries of the adjacency matrix for G, so that:

$$a(C_i) = \sum_{u \in C_i} \sum_{v \in \overline{C_i}} a_{u,v} \tag{3.4}$$

is the total number (or weight) of the edges incident with C_i.

The conductance of the whole graph, κ, is the minimum conductance over all its partition:

$$\kappa = \min_{C_i \in \mathscr{C}} K(C_i). \tag{3.5}$$

The lower the value of κ the higher the partition quality.

Indeed, the algorithmic schema implemented by RDyn (see Algorithm 1, lines 8–11) can be easily generalized in order to apply arbitrary partition quality functions, thus changing the rationale behind the planted communities and their evolution (i.e. optimizing partition density instead of conductance: we will compare this two choices in Section 4).

3.3.1 *Detecting merge and split.* At the end of each iteration, RDyn computes the current partition quality, $\widehat{\kappa}$, on the network partition imposed by the nodes communities allocation in order to decide whether the communities are stable enough to be returned as a valuable ground-truth or not.

Definition 3.3 (Stable partition/iteration) Given a partition P_i observed at the end of an iteration i, a partition quality threshold κ and the partition quality $\widehat{\kappa}$ computed on P_i we call i *stable* iff $\widehat{\kappa} \leq \kappa$. Symmetrically, we call *stable* an iteration i associated to a *stable* partition P_i.

Once defined when a partition is considered a valid output for RDyn, we need an algorithmic schema that introduces community dynamics and takes care of interpolating from the current stable state to the following one (Algorithm 1, lines 8–11). Reached a stable state, one or more split and/or merge actions can be planted: we regulate them by applying the following patterns.
Merge action:

i) two communities that will be involved in the merge event are chosen through a sample biased by the reciprocal of community sizes, giving priority to the merging of small communities;
ii) all the nodes belonging to the selected communities are marked as belonging to the same set: bridges among them are then labelled as internal edges.

Split action:

i) a community is chosen through a sample biased on community sizes, giving priority to the splitting of bigger communities.
ii) two disjoint sets of nodes are extracted from the selected community and labelled accordingly: interactions connecting nodes belonging to different communities are labelled as bridges.

The policies described to identify the communities involved in updates (both in the split and merge scenarios) are devised so to capture the intuition that it is more likely that bigger communities can fall apart w.r.t. small ones and that, symmetrically, small communities are more likely to merge than bigger ones. Indeed this choice, in the long run, tends to uniform the community size distribution. Once updated the assignment of nodes to communities, subsequent iterations of RDyn will automatically converge to a novel partition of sufficiently high quality to being returned as stable ground-truth. Convergence is due to the defined linking strategies: indeed, intra-community links are stable than inter-community ones (higher renewing probability, ν): this causes the rapid vanishing of bridges that connects logically separated node sets that once belongs to the same community (split) as well as the rising of intra-community edges among once disconnected communities.

Indeed, in order to avoid instability, merges and splits can be logically planned only when the network has reached a stable community state (i.e. previous community actions have been completed). Moreover, multiple splits and merge can be planned at the same time under the constraint that there is no overlap among the communities they target. Ideally, RDyn execution will follow a pattern similar to the one exemplified in Fig. 1: unstable iterations that fill the gap between stable ones represent the time window needed to complete the planned merge and split actions and are characterized by *expand* and *contract* actions of the communities selected to be *merged* or *split*.

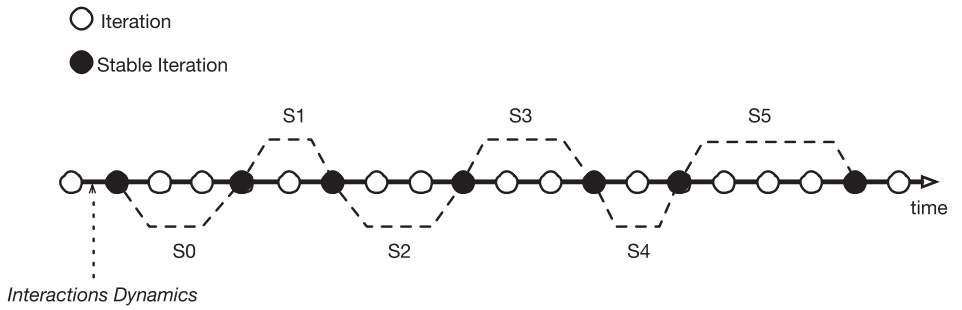

FIG. 1. RDYN execution timeline: ground-truth communities are generated only during stable iterations (black circles). Interactions between two consecutive stable iterations compose a snapshot (here identified with {S0,…,S5}). Interaction dynamics (as well as community ones) happens between consecutive iteration. Due to their definitions stable iterations are not bounded to appear with fixed displacement.

4. Benchmark: RDyn and NF1

When designing ground truth testing environments it is mandatory in order to assess the effectiveness of an algorithm to measure the degree of resemblance its output are able to provide w.r.t. a golden standard. In Section 3, we introduced our dynamic network generator RDYN: in order to propose a complete benchmark for the evaluation of CD algorithms, both for static and dynamic networks, here we discuss a novel metric, NF1 an extension of a quality function proposed in [37].

Given a community set X produced by a CD algorithm and a ground truth community set Y, for each community $x \in X$ we label its nodes with the ground truth community $y \in Y$ they belong to. Each community x is then matched with the ground truth community with the highest number of labels in the algorithm community. This procedure generates (x, y) pairs having the highest homophily between the node labels in x and all the ground truth communities. In [37], the authors measure the quality of the mappings by the two following measures:

- *Precision*: the percentage of nodes in x labelled as y, computed as

$$P = \frac{|x \cap y|}{|x|} \in [0, 1]. \tag{4.1}$$

- *Recall*: the percentage of nodes in y covered by x, computed as

$$R = \frac{|x \cap y|}{|y|} \in [0, 1]. \tag{4.2}$$

Given a pair (x, y) the two measures describe the overlap of their members: a perfect match is obtained when both *precision* and *recall* are 1. The identified pairs set describe a many-to-one mapping: multiple communities in X can be connected to a single ground truth community in Y. This policy enables the adoption of the proposed methodology both in the case of algorithms producing crisp partitions or algorithm producing overlapping communities. Moreover, analyzing the *precision* and *recall* of each pair it is possible to detect both underestimations and overestimations made by the adopted algorithm.

In order to provide a complete quality indicators, in [37] *precision* and *recall* are combined into their harmonic mean obtaining the $F1$-measure, a concise quality score for the individual pairing:

$$F1 = 2\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \tag{4.3}$$

Given a network, the $F1$ score is then averaged among all the identified pairs in order to summarize the overall correspondence between the algorithm community set and ground truth community set.

Here we propose an extension of the average $F1$ score: in order to mitigate the issues related to coverage and redundancy of communities while assessing the final matching quality, we define a normalized version of such metric, namely $NF1$.[3] In particular, defined Y_{id} the subset of community in Y matched by community in X, we can define *Coverage* as:

$$Coverage = \frac{|Y_{id}|}{|Y|} \in [0, 1]. \tag{4.4}$$

Coverage identifies the percentage of communities in Y that are matched by at least an object in X. Redundancy instead can be defined as:

$$Redundancy = \frac{|X|}{|Y_{id}|} \in [1, +\infty). \tag{4.5}$$

Redundancy is minimized when no conflicting matches exist among the communities in X and the ones in Y_{id}. Finally NF1 can be defined as:

$$NF1 = \frac{F1 * \text{Coverage}}{\text{Redundancy}} \in (0, 1] \tag{4.6}$$

NF1 is maximized when: (i) the average F1 is maximal (perfect match), (ii) the community in X provide a complete coverage for the ones in Y and (iii) the redundancy is minimized (i.e. each community in X is matched with a distinct community in Y).

As shown in [37], it is possible to compute F1 (and thus NF1) paying a linear complexity in the size of the community set X. The reduced complexity makes NF1 a suitable alternative to NMI [27] (whose complexity has shown to be $\mathcal{O}(|X|^2)$). Moreover, as discussed in [28, 31], NMI is not stable while comparing overlapping partitions with non-overlapping ones while NF1 does not suffer such limitation.

5. Experiments

In this section, we analyze the dynamic network structures generated by RDyn. In particular, in Section 5.1 we compare several CD algorithms (for both static and dynamic graphs) on networks generated by RDyn and LFR so to highlight the major differences among the two benchmarks. Later on, in 5.2, we

[3] Code available at: https://github.com/GiulioRossetti/f1-communities

discuss scalability of the proposed generative model and provide a characterization of the communities and networks it builds.

5.1 RDyn *vs.* LFR

As we have discussed in Section 2, whenever a new community algorithm needs to be tested against a synthetic benchmark the preferential choice within the state of art lies on LFR. We highlighted that RDyn, differently from LFR, is specifically tailored to the generation of dynamic graphs enriched with evolving ground truth communities: however, our approach can also be used to evaluate, on a single snapshot, classical static CD algorithms.

In order to compare the two benchmarks, we decided to evaluate three dynamic and three static CD algorithms on them, namely:

Static approaches:

- Demon [12]: is an incremental algorithm based on the analysis of ego networks from which micro communities are identified and merged. The communities are extracted by using a bottom-up procedure: each node gives the perspective of the communities surrounding it and then all the different perspectives are merged together in an overlapping structure.
- Infohiermap [38]: is one of best performing hierarchical non-overlapping clustering algorithms for community detection studied to optimize community conductance. The graph structure is explored with a number of random walks of a given length and with a given probability of jumping into a random node. The underlying intuition is that random walkers are trapped in a community and exit from it very rarely.
- Louvain [6]: is an heuristic method based on modularity optimization and it is proven to be fast and scalable on large-scale networks. The modularity optimization is performed in two steps. First, the method searches for 'small' communities by optimizing modularity locally. Second, it aggregates nodes belonging to the same community and builds a new network whose nodes are communities. These steps are repeated iteratively until a maximum modularity is obtained, producing a complete non-overlapping partitioning of the graph.

Dynamic approaches:

- D-GT [21]: is a game-theoretic approach for community detection in dynamic social networks: each node is treated as a rational agent who periodically chooses from a set of predefined actions in order to maximize its utility function. The community structure of a snapshot emerges after the game reaches Nash equilibrium; the partitions and agent information are then transferred to the next snapshot. D-GT attempts to simulate the decision-making process of the individuals creating communities, rather than focusing on statistical correlations between labels of neighbouring nodes.
- iLCD [10]: is an algorithm for the detection of overlapping communities in dynamic networks. It is not based on the modularity, but, on the contrary, on the idea that communities are defined locally (intrinsic communities).
- Tiles [36]: is an online algorithm that dynamically tracks overlapping communities in an edge stream graph following local topology perturbations. It applies constrained label propagation.

In order to evaluate the selected methods on the two benchmarks, we generated several networks having different configurations of the required parameters and analyze how the average quality of the identified communities vary with them. In particular, we define the following experimental setup:

TABLE 2 *Experimental parameters settings. The κ value refers to the minimum conductance/density required to identify a partition as stable*

Parameter	Values
ν	[0.5, 0.6, 0.7]
p_{in}	[0.6, 0.7, 0.8, 0.9]
κ (density)	[0.1, 0.15, 0.2, 0.25, 0.3]
κ (conductance)	[0.3, 0.35, 0.4, 0.45, 0.5, 0.55]

LFR:

- we generated 90 networks of 1000 nodes each varying μ, the mixing coefficient, and d, the network density, respectively in the range [0, .9] and [0, 1] (steps of .1);
- for each algorithm we computed the average quality score for each value of μ, varying the density value.

RDYN:

- we generated 1000 RDYN iterations for a set of networks of 1000 nodes each varying the parameters as shown in Table 2[4];
- for each algorithm we computed the average quality score for each value of κ (the community quality threshold), varying the remaining parameters. We decided to fix the number iteration to 1000 for each of the RDYN instantiation. Moreover, in order to better characterize the obtained results, we repeated the procedure twice employing respectively conductance and density as quality functions during the network generation phase.

Moreover, for both models we fix the exponent of power-law distributions for node degree and community size to 2.5 (as done in the [26]). Figure 4(a–f) shows the trends of the NF1score for the six algorithms on the two benchmarks: in order to better discuss the obtained results we separated the plots by benchmark: RDYN (conductance Fig. 2(a and d) , density Fig. 2(b and e) and LFR Fig. 2(c and e)), as well as for algorithm type (static/dynamic).

Two different peculiarities emerge from the plots: (i) RDYN and LFR propose different rankings among the tested algorithms, and (ii) RDYN benchmark appears to be, on average, more challenging than LFR (even for high community quality, i.e. low conductance/high density). Obviously, even though LFR and RDYN guarantee almost the same graph characteristics (i.e. power law degree and community size distribution) the local topologies they generate can highly differ due to the rationale behind their construction. In order to prove that the communities generated by RDYN are easily identifiable by methods that optimize the same quality function it employs we introduce in our experimental settings two static algorithms that search for, respectively, low conductance cuts (e.g. INFOHIERMAP) and dense communities (e.g., DEMON). As shown in Fig. 2(d–e), INFOHIERMAP is able to reach, and steadily maintain, a high NF1 on RDYN when conductance is used as quality function while generating communities, conversely it ranked

[4] When conductance is used as quality function, the generated networks are 360: 300 when considering density. For each parameter configuration, multiple RDYN instantiations were performed.

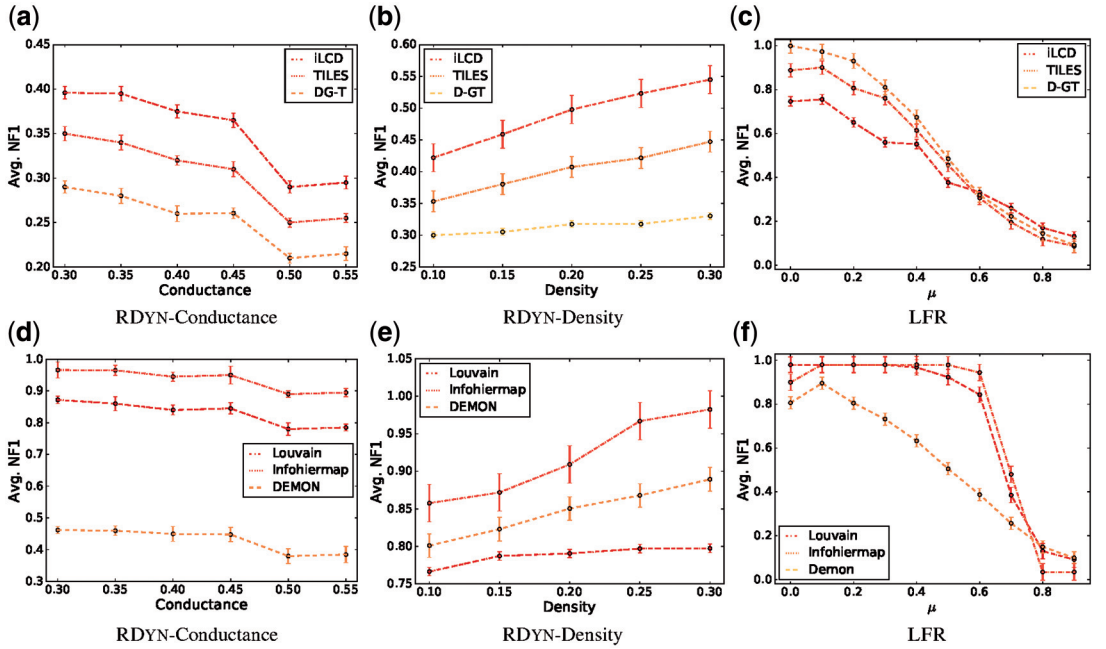

FIG. 2. Comparison of the NF1 of the CD and DCD algorithms on RDYN and LFR. In the first row are reported the NF1 trend line for the chosen DCD algorithm while varying the community quality, in the second the same visualization is shown for the tested CD approaches. Error bars highlight standard deviations.

third across the static methods when density is the community generation criterion. On the other hand, DEMON is not able to reach the best performances in its category for neither density nor conductance. This result, along with the ones of TILES and iLCD, is primarily due to the type of topologies the algorithm search for: indeed, RDYN generates non-overlapping, crisp, partitions thus penalizing approaches that allow a node to belong to different communities.

While analysing dynamic approaches we observe how D-GT is able to guarantee the highest performances on LFR but not in RDYN. This result is due to the nature of the algorithm itself: contrarily from its direct competitors, D-GT works on snapshot graphs, identifying a community set for each temporal observation, thus reducing the dynamic analysis to a sequence of static ones. The fact that both TILES and iLCD are able to reach higher NF1 than D-GT on RDYN is expected since they are fully exploiting the dynamics of network formation expressed by the model. Conversely, while applied on static graphs – as the ones generated by LFR–the two approaches suffer the lack of an explicit edge ordering thus reducing their effectiveness in retrieving the hidden ground truth w.r.t. to more classical approaches as D-GT or the static ones.

5.2 RDYN: *network and community characterization*

In Section 3, we described the rationale that regulates network and communities evolution in RDYN. Since the aim of RDYN is to provide support for dynamic community analysis in this section we highlight the most relevant properties of the partitions as well as of the dynamic networks it produces. To do

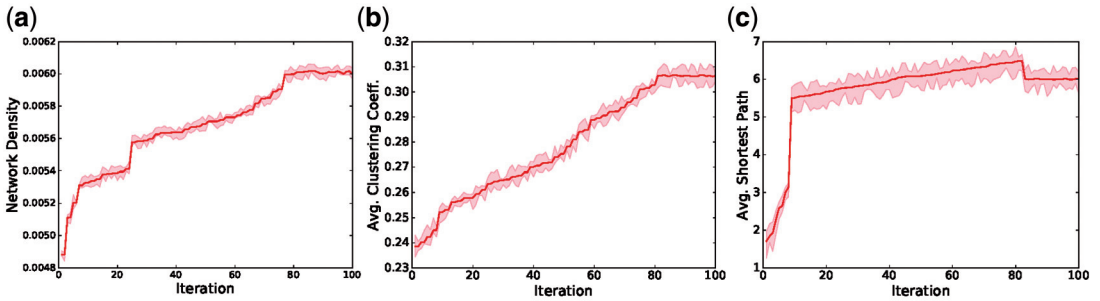

FIG. 3. Evolution of RDYN synthetic networks characteristics. The trend lines capture the values, averaged across all RDYN instantiations, of network density, average clustering coefficient and path length along with their interquartile ranges. Only the first 100 iterations are reported since after that the trends reach, and constantly maintain, a stable state.

so we exploit the synthetic dynamic networks generated using conductance as quality function (RDYN parameters in Table 2).

Figure 3 shows the average trend of networks density, shortest path length and global clustering coefficient for all RDYN executions (along with the interquartile range). We can observe that all the trends appear to slowly reach an asymptotic value after a first growing phase: in the plots, we report only the first 100 iterations since after that all the trend lines stabilizes. Indeed, network density can reach at most $\frac{|E|}{|V|(|V|-1)/2}$: since we implicitly fix the maximum number of edges per node adopting a planted degree distribution, once network density come close to such upper bound it can only experience to small fluctuations. Moreover, the policy used to preferentially rewire edges within/across communities—regulated by p_{in}—as well as the enforcing of triadic closures make the trends of clustering coefficient and average shortest path behave alike.

Moving to the analysis of the generated communities, we firstly focus on the impact of κ, the quality threshold, on the obtained partitions. Figure 4(a) highlights that a raise in κ value causes an increase in the average number of generated community actions during the 1000 iterations analyzed. Indeed, this phenomenon is easily explainable: increasing the conductance threshold lowers the minimum partition quality requested in order to identify a stable state and thus plan future community actions. A related pattern emerges while comparing how the average partition modularity relates to the conductance threshold, Fig. 4(b). Even in this case, we observe a reasonable effect: lowering the required partition quality lowers the average modularity score. Conversely, in Fig. 4(c) we can observe how κ does not particularly affect the average density of the identified communities.

In order to better understand how the graph evolution changes when RDYN is configured with different parameters values in Fig. 4(d and e) we extrapolated the trend of, respectively, average number of stable iterations and average modularity while varying both κ and p_{in}. We observe that the number of stable iterations is tied not only to the imposed quality threshold but also to the p_{in} value: the higher the ratio among intra/inter edges required for each node the better separated and well defined the communities generated. This correlation implies a higher chance for unstable iteration to rapidly converge to stable states after merge/split actions. Moreover, p_{in} affect the speed in modularity drop for increasing value of κ: increasing the intra-community edge probability we get a more pronounced the modularity drop.

Figure 4(f) shows the execution time of RDYN while varying the number of nodes within the graph (the number of iterations is always kept equals to the number of nodes). It is straightforward to notice

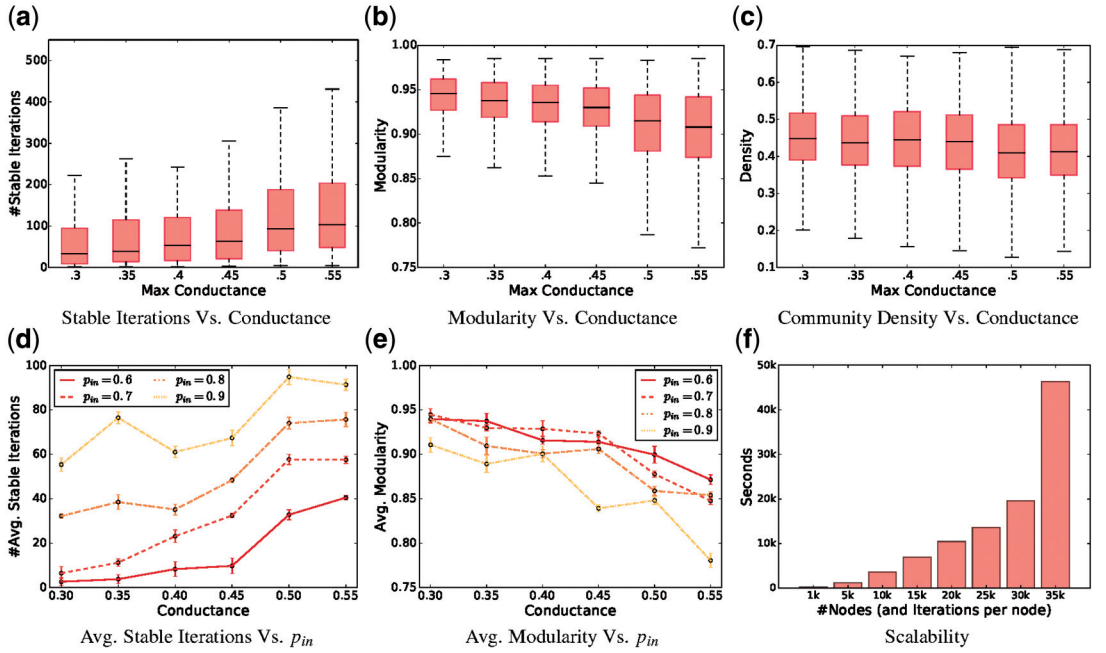

FIG. 4. RDYN ground truth planted community quality and execution time. In the first row is shown how, varying the quality threshold (conductance) the distribution of the number of stable iteration (a), modularity (b) community density and (c) change. In the second row are reported the trend for the number of average stable iteration (d) and modularity (e) while varying the quality threshold (conductance) and the intra-community edge probability. Finally, in (f) it is shown the relations between the RDYN execution time and size of the simulation (i.e. graph size and iterations number). In (d–f), error bars identify interquartile ranges.

that the size of the network (and, consequently the number of performed iterations) deeply affect the generation time: however, it should be noted that the actual result of a single RDYN instantiation is composed by: (i) an ordered sequence of edge insertion/removal, (ii) a set of stable network snapshots and (iii) a set of stable community ground truths. Indeed, this detailed output affects the execution time but enables the adoption of RDYN to test both static CD (as well as snapshot based DCD) algorithms and DCD approaches using ordered interaction graph representation. Since the outputs provided by LFR and RDYN are not the same (the former does not handle network dynamics and generates a single snapshot graph with its ground truth partition) we omit a scalability comparison.

From our experiments emerges that RDYN parameters succeeded in modelling three different aspects of dynamic networks and communities:

- Community density and separation: increasing the p_{in} value the generated mesoscale structures tend to be connected by fewer bridges as well as to increase the number of edges within them;
- Interaction/communities stability: high ν values give rise to interactions that are more likely to be renewed, thus describing more stable topologies;
- Community quality and the number of stable iterations: tuning the κ threshold RDYN defines the overall quality of the communities generated during stable iterations. Moreover, the higher the quality required the lower the number of stable iterations.

Indeed, by tuning such parameters it is possible to describe benchmarks having different characteristics and complexity.

6. Discussion and conclusion

In this work, we proposed RDyn a dynamic network generator able to handle community dynamics. RDyn is designed to guarantee power law degree distributions of both node degree and community sizes and can be instantiated so to obtain planted communities that optimize different quality measures.

The proposed method firstly generates both the community size and degree distribution and use them to associate each node to a partition, then it introduces network dynamics by iteratively adding and removing edges. Once RDyn identifies a stable state, that is a state in which the quality of the planted partition is sufficiently high (w.r.t. a user-defined threshold) it outputs it as ground truth and generates community perturbations (split/merge events) by changing node-community associations of selected communities. Changing node-community associations has the effect of update the pre-existing planted partition thus describing a novel status toward which the network will converge due to the policies used to add/remove edges.

RDyn exposes four parameters (i.e. desired number of nodes, minimum partition quality, inter-intra community edge ratio, probability of interaction renewal) in order to allow the final user for a fine-grained definition of the desired network topology and dynamics. Indeed, the distributions used to model network topology can be assigned using results coming from previous researchers on real network topological characteristics (i.e. the power law degree exponents of both degree and community sizes as well as the interaction decay one): however, we design our model to be as general as possible leaving to the analyst the possibility to specify distribution that diverges from the ones implemented as default.The proposed model not only makes assumptions on the properties of the generated graph but also on its dynamics: this second dimension – which indeed increases RDyn complexity—makes our approach different from previous ones and explicitly tailored to benchmark a specific class of CD algorithms.

As we have discussed, dynamic CD, that is the task of identifying community evolutive patterns in dynamic networks, is a rising theme in complex network analysis. RDyn offers evolving and tunable-quality planted partitions that can be fruitfully used to benchmark approaches that search for communities in a dynamic network scenario. We analyzed the networks and communities generated by the RDyn varying its parameters, moreover we proposed a community matching quality score, namely NF1 (extended from [37]) thus describing a complete testing environment able to assess the effectiveness of dynamic CD algorithms w.r.t. synthetic evolving networks. Our experiments show that RDyn represents a more complex benchmark than LFR for both static and dynamic community detection algorithms. Moreover, we observed that p_{in} (the intra-community edge probability) plays a crucial role in defining both the number of stable states and modularity scores for the communities generated by RDyn (while maintaining fixed the number of total iterations and quality threshold): the higher the intra-community edge probability the higher the number of stable states, the lower the average modularity of the identified communities.

An interesting line of research that we propose to carry on involves the comparison of RDyn's synthetic graphs with real-world dynamic network datasets (as the ones available on the SocioPatterns platform[5]). This analysis will allow us to identify reliable parameter configurations so to better approximate the dynamics of specific categories of real world phenomena. Indeed, RDyn can be easily extended to simulate more fine-grained network dynamics (e.g. node appearance and vanishing) as well as to model

[5] http://sociopatterns.org/datasets/

more general scenarios (e.g. dynamic systems in which not all the nodes are involved in interactions during each iteration). Such extensions are not discussed in this work but represent a line of research we are currently developing with the aim of building a completely general purpose dynamic network benchmark. Moreover, we propose to extend RDyn in order to handle overlapping and hierarchical communities.

Funding

This work is funded by the European Community's H2020 Program under the funding scheme 'FETPROACT-1-2014: Global Systems Science (GSS)', grant agreement (# 641191) CIMPLEX 'Bringing CItizens, Models and Data together in Participatory, Interactive SociaL EXploratories' (https://www.cimplex-project.eu). This work is supported by the European Community's H2020 Program under the scheme 'INFRAIA-1-2014-2015: Research Infrastructures', grant agreement (#654024) *SoBigData: Social Mining & Big Data Ecosystem* (http://www.sobigdata.eu).

References

1. Adamic, L. A., Lukose, R. M., Puniyani, A. R. & Huberman, B. A. (2001) Search in power-law networks. *Phys. Rev. E.*, **64**, 046135.
2. Aiello, W., Chung, F. & Lu, L. (2000) A random graph model for massive graphs. *Proceedings of the thirty-second annual ACM symposium on Theory of computing.* New York, NY, USA: ACM, pp. 171–180.
3. Aynaud, T., Fleury, E., Guillaume, J-L. & Wang, Q. (2013) Communities in evolving networks: definitions, detection, and analysis techniques. *Dynamics on and of Complex Networks*, vol 2. doi:10.1007/978-1-4614-6729-8.
4. Barabasi, A. L. & Albert, R. (1999) Emergence of scaling in random networks. *Science.*, **286.5439**, 509–512.
5. Berlingerio, M., Coscia, M. & Giannotti, F. (2009) Mining the temporal dimension of the information propagation. *International Symposium on Intelligent Data Analysis.* Heidelberg, Berlin: Springer, pp. 237–248.
6. Blondel, V. D., Guillaume, J-L., Lambiotte, R. & Lefebvre, E. (2008) Fast unfolding of communities in large networks. *J. Stat. Mech. Theory E.*, **2008.10**, P10008.
7. Bollobas, B. (2013) *Modern Graph Theory*, vol. 184. New York: Springer Science & Business Media.
8. Brandes, U., Gaertler, M. & Wagner, D. (2003) *Experiments on Graph Clustering Algorithms*. Heidelberg, Berlin: Springer.
9. Cazabet, R. & Amblard, F. (2014) Dynamic community detection. *Encyclopedia of Social Network Analysis and Mining*. New York: Springer, pp. 404–414.
10. Cazabet, R., Amblard, F. & Hanachi, C. (2010) Detection of overlapping communities in dynamical social networks. *Second International Conference on Social Computing (SocialCom)*. IEEE, pp. 309–314.
11. Condon, A. & Karp, R. M. (2001) Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms.*, **18.2**, 116–140.
12. Coscia, M., Rossetti, G., Giannotti, F. & Pedreschi, D. (2014) Uncovering hierarchical and overlapping communities with a local-first approach. *ACM Transactions on Knowledge Discovery from Data (TKDD).*, **9**, 6.
13. Danon, L., Diaz-Guilera, A. & Arenas, Alex, A. (2006) The effect of size heterogeneity on community identification in complex networks. *J. Stat. Mech. Theory E.*, **2006.11**, P11010.
14. Erdos, P. & Renyi, A. (1959) On random graphs. *Publ. Math. Debrecen.*, **6**, 290–297.
15. Fortunato, S. (2010) Community detection in graphs. *Phys. Rep.*, **486**, 75–174.
16. Fan, Y., Li, M., Zhang, P., Wu, J. & Di, Z. (2007) Accuracy and precision of methods for community identification in weighted networks. *Physica A.*, **377.1**, 363–372.
17. Gaertler, M., Görke, R. & Wagner, D. (2007) Significance-driven graph clustering. *Algorithmic Aspects in Information and Management*. Heidelberg, Berlin: Springer-Verlag, pp. 11–26.

18. GIRVAN, M. & NEWMAN, M. E. J. (2002) Community structure in social and biological networks. *Proceedings of the national academy of sciences*., **99.12**, 7821–7826.

19. GREENE, D., DOYLE, D. & CUNNINGHAM, P. (2010) Tracking the evolution of communities in dynamic social networks. *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, pp. 176–183.

20. GUIMERÀ, R., SALES-PARDO, M. & AMARAL, L. A. N. (2007) Module identification in bipartite and directed networks. *Phys. Rev. E.*, **3**., 036102

21. ALVARI, H., HAJIBAGHERI, A. & SUKTHANKAR, G. (2014) Community detection in dynamic social networks: A game-theoretic approach. *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE/ACM, pp. 101–107.

22. HARTMANN, T., KAPPES, A. & WAGNER, D. (2016) Clustering evolving networks. *Algorithm Engineering*. Springer International Publishing. pp. 280–329.

23. HOLME, P. & KIM, B. J. (2002) Growing scale-free networks with tunable clustering. *Phys. Rev. E.*, **65.2**, 026107.

24. JEONG, H., MASON, S. P., BARABASI, A. L. & OLTVAI, Z. N. (2001) Lethality and centrality in protein networks. *Nature*, **411**, 41–42.

25. KUMPULA, J. M., ONNELA, J. P., SARAMAKI, J., KASKI, K. & KERTESZ, J. (2007) Emergence of communities in weighted networks. *Phys. Rev. Lett.*, **99.22**, 228701.

26. LANCICHINETTI, A. & FORTUNATO, S. (2009) Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E.*, **1**, 016118.

27. LANCICHINETTI, A., FORTUNATO, S. & RADICCHI, F. (2008) Benchmark graphs for testing community detection algorithms. *Phys. Rev. E.*, 046110.

28. LANCICHINETTI, A., FORTUNATO, S. & KERTESZ, J. (2009) Detecting the overlapping and hierarchical community structure in complex networks. *New J. Phys.*, **11.3**, 033015.

29. LESKOVEC, J., KLEINBERG, J. M. & FALOUTSOS, C. (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. New York, NY, USA: ACM, pp. 177–187.

30. LIN, Y. R., CHI, Y., ZHU, S., SUNDARAM, H., & TSENG, B. L. (2008) Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. *Proceedings of the 17th international conference on World Wide Web*. ACM, pp. 685–694.

31. MCDAID, A. F., GREENE, D. & HURLEY, N. J. (2011) Normalized mutual information to evaluate overlapping community finding algorithms. arXiv preprint arXiv:1110.2515

32. PASTA, M. Q. & FARAZ, Z. (2016) Network generation model based on evolution dynamics to generate benchmark graphs. arXiv https://arxiv.org/abs/1606.01169.

33. MUSIAL, K. & KAZIENKO, P. (2012) Social networks on the internet. *World Wide Web J.*, doi:10.1007/s11280-011-0155-z

34. PALLA, G., BARABÁSI, A-L. & VICSEK, T. (2007) Quantifying social group evolution. *Nature*. doi:/10.1038/nature05670.

35. PASTA, M. Q., JAN, Z., SALLABERRY, A. & ZAIDI, F. (2013) Tunable and growing network generation model with community structures. *Third International Conference on Cloud and Green Computing (CGC)*. IEEE, pp. 233–240.

36. ROSSETTI, G., PAPPALARDO, L., PEDRESCHI, D. & GIANNOTTI, F. (2016) Tiles: an online algorithm for community discovery in dynamic social networks. *Mach. Learn.*, 1–29. doi:10.1007/s10994-016-5582-8.

37. ROSSETTI, G., PAPPALARDO, L. & RINZIVILLO, S. (2016) A novel approach to evaluate community detection algorithms on ground truth. *Complex Networks VII*. Switzerland: Springer International Publishing, pp. 133–144.

38. ROSVALL, M. & BERGSTROM, C. T. (2011) Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*., **6.4**, e18209.

39. SAWARDECKER, E. N., SALES-PARDO, M. & AMARAL, L. A. N. (2009) Detection of node group membership in networks with group overlap. *Eur. Phys. J. B.*, **67.3**, 277–284.

40. STEHLE, J., BARRAT, A. & BIANCONI, G. (2010) Dynamical and bursty interactions in social networks. *Phys. Rev. E.*, **81**, 035101.

41. WATTS, D. J. & STROGATZ, S. H. (1998) Collective dynamics of small-world networks. *Nature*, **393**, 440–442.
42. XIE, W., TIAN, Y., SISMANIS, Y., BALMIN, A. & HAAS, P. J. (2015) Dynamic interaction graphs with probabilistic edge decay. *31st International Conference on Data Engineering (ICDE)*, IEEE, pp. 1143–1154.
43. XIE, Z., LI, X. & WANG, X. (2007) A new community-based evolving network model. *Physica A.*, **384.2**, 725–732.
44. XU, X-J., ZHANG, X. & MENDES, J. F. F. (2009) Growing community networks with local events. *Physica A.*, **388.7**, 1273–1278.
45. YANG, J. & LESKOVEC, J. (2015) Defining and evaluating network communities based on ground-truth. *Knowl. Inform. Syst.*, **42**, 181–213.
46. ZAIDI, F. (2013) Small world networks and clustered small world networks with random connectivity. *Soc. Network Anal. Min.*, **3.1**, 51–63.
47. ZAIDI, F., PASTA, M. Q., SALLABERRY, A. & MELANCON, G. (2015) Social ties, homophily and extraversion–introversion to generate complex networks. *Soc. Network Anal. Min.*, **5.1**, 1–12.
48. ZHAO, K., STEHL, J., BIANCONI, G. & BARRAT, A. (2011) Social network dynamics of face-to-face interactions. *Phys. Rev. E.*, **83**, 056109.
49. ZHOU, X., XIANG, L. & WANG, X. (2008) Weighted evolving networks with self-organized communities. *Commun. Theor. Phys.*, **50.1**, 261.