ISCTE IUL
REPOSITÓRIO
INSTITUTO UNIVERSITÁRIO DE LISBOA

Repositório ISCTE-IUL

Deposited in *Repositório ISCTE-IUL*:
2020-03-12

Deposited version:
Post-print

Peer-review status of attached file:
Peer-reviewed

Citation for published item:
Pereira, L. R., Louçã, J. & Lopes, R. J. (2019). Syntgen: a system to generate temporal networks with user specified topology. Journal of Complex Networks. N/A

Further information on publisher's website:
10.1093/comnet/cnz039

Publisher's copyright statement:
This is the peer reviewed version of the following article: Pereira, L. R., Louçã, J. & Lopes, R. J. (2019). Syntgen: a system to generate temporal networks with user specified topology. Journal of Complex Networks. N/A, which has been published in final form at https://dx.doi.org/10.1093/comnet/cnz039. This article may be used for non-commercial purposes in accordance with the Publisher's Terms and Conditions for self-archiving.

Use policy

Creative Commons CC BY 4.0
The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a link is made to the metadata record in the Repository
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Serviços de Informação e Documentação, Instituto Universitário de Lisboa (ISCTE-IUL)
Av. das Forças Armadas, Edifício II, 1649-026 Lisboa Portugal
Phone: +(351) 217 903 024 | e-mail: administrador.repositorio@iscte-iul.pt
https://repositorio.iscte-iul.pt

IMA Journal of Complex Networks (2019) Page 1 of 24
doi:10.1093/comnet/xxx000

Syntgen: A system to generate temporal networks with user specified topology

LUIS RAMADA PEREIRA*
ISTAR Instituto Universitário de Lisboa (ISCTE - IUL) Lisbon, Portugal
*Corresponding author: m4472@iscte-iul.pt

RUI J. LOPES
IT-IUL Instituto de Telecomunicações (ISCTE - IUL) Lisbon, Portugal

AND

JORGE LOUÇÃ
ISTAR Instituto Universitário de Lisboa (ISCTE - IUL) Lisbon, Portugal

[Received on 4 March 2019]

Network representations can help reveal the behavior of complex systems. Useful information can be derived from the network properties and invariants, such as components, clusters or cliques, as well as from their changes over time. In the last few years the study of temporal networks has progressed markedly. The evolution of clusters of nodes (or communities) is one of the major focus of these studies. However, the time dimension increases complexity, introducing new constructs and requiring novel and enhanced algorithms. In spite of recent improvements, the relative scarcity of timestamped representations of empiric networks, with known ground truth, hinders algorithm validation. A few approaches have been proposed to generate synthetic temporal networks that conform to static topological specifications while in general adopting an ad-hoc approach to temporal evolution. We believe there is still a need for a principled synthetic network generator that conforms to problem domain topological specifications from a static as well as temporal perspective. Here we present such a system. The unique attributes of our system include accepting arbitrary node degree and cluster size distributions and temporal evolution under user control, while supporting tunable joint distribution and temporal correlation of node degrees. Theoretical contributions include the analysis of conditions for "graphability" of sequences of inter and intra cluster node degrees and cluster sizes and the development of a heuristic to search for the cluster membership of nodes that minimizes the shared information distance between clusterings. Our work shows that this system is capable of generating networks under user controlled topology with up to thousands of nodes and hundreds of clusters with strong topology adherence. Much larger networks are possible with relaxed requirements. The generated networks support algorithm validation as well as problem domain analysis.

Keywords: Graph Algorithms, Network flows, Clustering, Temporal Networks, Topology

1. Introduction

Networks are all around us: computer, telecommunication, biological and social systems are just a few examples of systems of entities that interact and relate to one another in some specifiable way, producing identifiable phenomena. Graph theory, which had its origins in the 18th century when Leonard Euler published his "Seven Bridges of Königsberg" problem and its negative solution [12], is the basis of the field of study that has become network science. Network science is concerned with understanding

© The author 2019. Published by Oxford University Press on behalf of the Institute of Mathematics and its Applications. All rights reserved.

networked systems, describing their micro, meso and macro scale attributes and helping us predict their behavior. Many networks exhibit groups of nodes that are more closely interconnected amongst themselves than with the rest of the network. These groups, referred to as clusters in graph theory or communities in network science, are usually of particular interest to network researchers. They may have an over-sized impact on the network behavior and their identification is often highly useful.

From its origin in graph theory, network science has focused on static networks, that is, networks "frozen in time" with link permanence. However, real world systems are rarely static: links on webpages are added and removed everyday in the world wide web, amino acid interaction for protein folding occurs over time, friendships are created, age, wither and renew. This realization led to major efforts to extend existing science into temporal networks, with several authors proposing approaches that embed time specific attributes. Communities are no exception, and several constructs have been proposed to characterize the way a community develops over time. Although some of these constructs are problematic since they cannot be derived solely from the network structure, they serve as a base that allow us to build a commonly accepted vocabulary that helps advance this field of study.

Time stamped data of empiric systems with known ground truth about communities does not abound. As extensively discussed elsewhere [25] even the concept of community membership is not without its challenges. This makes it more difficult to test systems that effectively recover community node membership over time. Having a system that generates a temporal network under user specified topology, with known ground truth, can help alleviate these challenges. Syntgen[1], as described in this article, is such a system.

The input required by Syntgen at each time step is a multiset of community sizes and a bijection of two n-tuples representing sequences of total and intra-community node degrees. Optionally the user can specify preferences for joint node degree distributions, temporal node degree correlation and a set of nodes to be eliminated at each time transition.

We developed a method which Syntgen uses to test user specifications for graphability and, if successful, generate a compliant temporal network. The user does not specify node membership or edges, these are generated by the system. As there is randomness in the process of network construction, both on node community membership as well as in the network wiring, the same specifications will not typically generate the same network. However, they should asymptotically converge to the same average topology.

The user can loosely control the dynamics of the network by changing its input at chosen time steps, with new nodes created and others killed to satisfy input specifications. Changing correlation and joint distribution parameters will also impact the wiring of the network.

We provide example sequence generators that sample power laws, exponential and binomial distributions, all of which have been found in empirical networks [26]. These generators include parameters that specify community maximum and minimum size, maximum and minimum node degree, distribution rate parameters and a ratio (r) of intra to total degree, which can be fixed or Bernoulli distributed with $\mathbb{P} = r$. The user can use or adapt these generators or provide their own. Obviously, although the system will assign nodes to communities, these are only meaningful if the ratio of intra links to total links is sufficiently high. This ratio varies depending on the network structure and on the cardinalities of the communities. Larger communities are less stringent with their requirements. A thorough discussion can be found in page 11 of [14].

With this input, Syntgen outputs a temporal network with known ground truth of its community structure for every time interval. To minimize network changes beyond those specified by the user,

[1] code available on request from the authors

Syntgen tries to determine the node community membership across time steps that results in the shortest shared information distance between clusterings. This is an NP-Hard problem for which we develop an appropriate heuristic.

Our system works for simple networks. Syntgen generates temporal networks with no self loops or multi-edges, non weighted and undirected, with no community membership overlap, with no isolated nodes, in snapshot mode. A new instance is generated at each time step and the overall temporal network is the sequence of generated snapshots. It would be possible to extend this model to a truly continuous streaming network, although a principled approach for node and edge activation would need to be devised to enforce node degree and community size affinities.

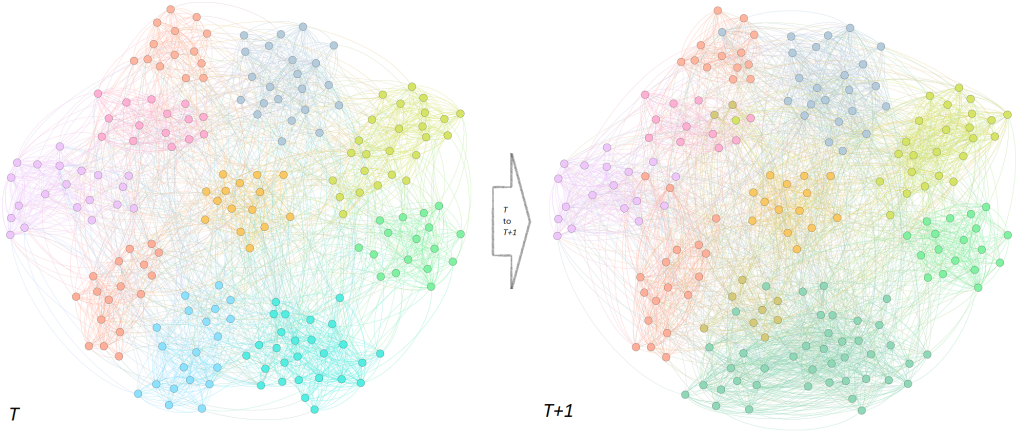


FIG. 1: **Time consecutive slices of a dynamic network as generated by Syntgen**
This example is color coded according to community and has ≈ 200 nodes, $10-9$ communities, with multiple community events. A full description can be found in section 3.2.5.

We believe Syntgen, as a temporal synthetic network generator, is unique in creating networks with arbitrary community sizes and node degree distributions and providing ways to control node joint degree distribution and node degree temporal correlation.

In the remainder of this document we review in section 2 other work related to the function and objective of our system. In section 3 we start by describing the general flow of the system, its modules and functionality, followed in section 3.2 by a detailed description of the approach we took to generate a network snapshot respecting the user topology. How we aim to reduce spurious noise when evolving the communities at every timestep is covered in section 3.3, and we conclude with 3 additional sections covering experiments, conclusion and future work.

A note on terminology conventions: In this subject area a vast array of terms are used to describe very similar concepts, like communities vs clusters, or partitions vs clusterings. Throughout this document, we adopt the following terminology and symbol conventions:

- ”Community” refers to groups of nodes more tightly connected amongst themselves than to the rest of the network (in lieu of terms like cluster, or partition)

- ”Clustering” refers to the splitting of a network into communities (in lieu of partition). See formal definition in 2.1.

- ”Temporal” is an attribute of a network that changes over time (in lieu of dynamic or evolving)
- ”Nodes and Links”: Links are connections between nodes at the same time step. Nodes only exist if they connect. There are no isolated nodes in Syntgen.
- We call the movement of nodes between communities across time steps, ”Node flow”.
- We define ”Graphability” as the property of sequences of community sizes and bijective node total and intra degrees that enable their graphical representation.
- We denote sets by an uppercase letter and individual elements by the corresponding lower case letter, optionally subscripted for identification. Frequently used sets and variables have their own dedicated symbol as per table 1.

Table 1: **Symbol convention**

Symbol	Definition
C	Clustering or community set, optionally with a superscript to indicate time step
D	Degree sequence (or ordered total degree sequence, depending on context, bijection with E, optionally with a superscript to indicate community membership).
E	Intra degree sequence (bijection with F), optionally with a superscript to indicate community membership.
F	Inter degree sequence , with $n = \|D\| = \|E\| = \|F\|$ and $f_i + e_i = d_i \; \forall (1 \leqslant i \leqslant n)$, optionally with a superscript to indicate community membership.
G	Network, optionally with a superscript to indicate time step
L	Link set
O	Kill node set (user specification)
S	Multiset of community sizes
T	Time step
U	Flow of nodes between time steps
V	Node set

2. Related Work

Work related to Syntgen falls into two categories:

- network science, theorems and algorithms that supported the development of our system
- prior systems that have been developed with similar or related desiderata.

In the first category we cover clustering similarity and community lifecycle events, and in the second, benchmarks for community detection algorithms and other temporal community network generators.

2.1 *Benchmarks for community detection algorithms*

Authors in [21] have drawn our attention to the fact that community detection algorithms that perform well in a given network topology may be less accurate in a different topology. Prior to their work, algorithms for community detection were validated mostly against the Girvan-Newman benchmark [25]. This a stochastic block model that only deviates from a typical random Erdős–Rényi model by the introduction of a tunable parameter specifying the probabilities of intra and inter community links, transforming the network from a pure random network to a random network of random networks (the communities). From experience we know that empirical networks do not generally follow this model. As an example, most networks generated by preferential attachment, that is networks where new nodes attach to existing nodes with probability that is dependent on their degree [2], end up with long tail distributions of node degrees and community sizes, reasonably approximated by power law distributions.

The benchmark introduced in [21] generates networks with power law distributions of community sizes and node degrees, with tunable intra/total ratio (mixing parameter). This benchmark, commonly known by the authors initials (LFR), has been widely accepted and used to test community detection algorithms for static networks. For instance in [20, 32] a lengthy list of algorithms tested against this benchmark can be found.

2.2 *Comparing clusterings*

A clustering, in our context, is the partition of the set V of nodes of a network into disjoint communities, or formally

$$C = \{c_1, \cdots, c_k\} : (c_i \cap c_j = \varnothing \ \forall\, (1 \leqslant i, j \leqslant k \wedge i \neq j) \wedge \cup_{i=1}^{k} c_i = V \tag{2.1}$$

Comparing communities at successive timesteps is a key requirement to understand community evolution. Comparing clusterings, on the other hand, is critical in our system so that, after all the information required to construct the network at successive time steps is gathered, we can flow nodes resulting in the closest shared information distance between clusterings. Comparing clusterings is an open problem as there is no standard way of measuring the distance between them. Popular methods include several variations of node counting (like the Rand Index) and measures from information theory, like the normalized mutual information and variation of information (VI). A good survey of different methods can be found in [31]. We have selected VI, given its robustness, low computational complexity and the fact that it is a true metric [23].

2.3 *Temporal community graph generators*

There have been some proposals to generate synthetic temporal networks. In [15] the authors propose a generator for simple networks with a cyclic nature based on a variation of the stochastic block model. In [16] the authors have adapted the LFR benchmark [21], while introducing over time ad-hoc modifications to the network. In [28] the authors propose RDyn, a system to generate temporal networks respecting a power-law distribution of community sizes and node degrees with tunable clustering and injected lifecycle events that, while disrupting cluster quality, are subsequently re-balanced through re-wiring of node links.

These systems have obvious affinity with Syntgen. The new contributions introduced by Syntgen include:

- no prior specifications of node degree or community size distribution

- temporal evolution under user control by searching for the least noisy transition across time steps, i.e. reducing partition artifacts as a result of node flow
- support for joint distribution of node degrees and node degree temporal correlation

3. Syntgen: Description, challenges and contributions

Syntgen is a system to create temporal networks exhibiting community structure that changes over time. It is parametric and modular. The major modules are:

- User specifications. These fall into two separate categories: network topology and heuristics execution.
- Node degree and community size sequence generators. The system includes functions that sample parametric distributions for community size, intra and total node degree, but, as long as they are realizable, any sequences can be provided.
- Network module. Deals with all aspects of network creation, including node to community assignment, degree to node assignment and node to link assignment.
- Transition module. This module manages all aspects of temporal evolution, including heuristics for node flow between timesteps and community lifecycle determination.
- Output module. This module generates all output, both textual as well as machine readable for further analysis. In Table 2 we include a summary of all information generated.

Table 2: **Textual Output of Syntgen**

Content	Description
Contingency Matrix	Contingency matrix of communities across time steps
Assortativity Coefficient	Joint node degree distribution
Temporal Degree Correlation	Average Pearsons correlation index for the whole network
Variation of information	VI between clusterings across successive time steps

In the remainder of this section we present the basic algorithmic logic of Syntgen in 3.1, the challenges and solutions of building a static network according to user specifications in 3.2 and the problem of finding a node flow across time steps that maximizes clustering similarity in 3.3.

3.1 *Syntgen basic logic*

The general flow of Syntgen is a sequence of looping steps that produce network snapshots as time progresses. It basically follows algorithm 1.

Syntgen requires from the user at each time step the following graph invariants and parameters:

- a multiset of k positive integers $S = \{s_1, \cdots, s_k\}$, representing a sequence of community sizes
- a bijection of total and intra community degree sequences:
  - an n-tuple of positive integers $D = \{d_1, \cdots, d_n\}$ representing a sequence of node total degrees with $n = \sum_{i=1}^{k} s_i \wedge \sum_{i=1}^{n} d_i \in \{2n : n \in \mathbb{N}\}$

  - an n-tuple of positive integers $E = \{e_1, \cdots, e_n\}$ representing a sequence of node intra-community degrees with $e_i \leqslant d_i : \ 1 \leqslant \forall i \leqslant n$ and $\sum_{i=1}^{n} e_i \in \{2n : n \in \mathbb{N}\}$

- specifications for joint degree distribution and node degree correlation over time

- optionally, a set of nodes O to kill at a step boundary

The user can loosely control the dynamics of the network by changing S, D, E and O at each time step boundary. Depending on the sign of $\sum S^t - \sum S^{t+1} - |O|$ new nodes are implicitly born or additional nodes randomly killed. Correlation and joint distribution parameters have an impact on the wiring of the network. As the data per timestep is gathered, Syntgen executes the following actions:

- A bootstrap static network is built. The input elements are independent (with the exception of the number of nodes and the sum of community sizes, which must match) and it is up to the system to assign edges and nodes to communities. We provide parameter-based examples of functions that generate sequences which have been observed in empirical networks ([8, 26, 27]), sampled from discretized power laws, discretized exponential and binomial distributions, but the user is free to provide his own as adequate to their problem domain.
  Degree assortativity, a topology attribute that varies with the type of network (typically assortative for social, while dissortative for biological networks [24]), is also parameter driven allowing the user to request a random, weighted assortative or weighted dissortative network.
  To construct the network we use a modified version of the configuration model [9] in a similar approach to what is found in a popular benchmark for community detection in static networks [21], but developed independently and extended to support joint node degree distributions as described in 3.2.5.
  Obviously, not all input specifications are possible and we verify feasibility before generating the network. The problem of whether a given node degree distribution can be expressed as a graph has been covered extensively in the literature [7, 11, 29, 30], and theorems, like the Erdös-Gallai condition, can be expressed as an algorithm to test graph feasibility. However, with node degrees as tuples of inter and intra-cluster degrees, different conditions apply. We extended the Erdös-Gallai condition to address this problem, developing the corresponding algorithm to halt (or request new input for) the network generation in case input specifications are infeasible.

- After generating the bootstrap network (T_0 network) a T_1 network is generated, again according to user specifications. The user may select a different degree or community size sequences as well as make changes to the network at the end of T_0 (selecting nodes for deletion), according to the requirements of the temporal network to be generated. An additional parameter provides the user with the option of enforcing node degree assortativity across timesteps.
  The system then tries to find the closest possible clusterings between successive timesteps. We found the problem to be NP-hard and impossible to complete in a reasonable amount of time beyond a very small number of communities. To address the inherent complexity, we developed a heuristic based on a greedy anytime algorithm with taboo to search for a solution in an appropriate solution subspace. The objective is to reduce the amount of change (noise) to a minimum, reducing the necessary impact on user specifications. The solution will determine the flow of nodes between communities in timesteps T_0 and T_1.

- This process is repeated for a user-specified number (n) of time steps, evolving the network over a period from T_0 to T_n.

- At each time step the contingency matrix of node/community evolution is produced, and, at the end, the temporal network is created in a machine readable format for further analysis and visualization.

Algorithm 1 General flow of Syntgen. Steps 2,3,6,10 are implemented in the "Network" module. *Build Communities* assigns degrees to nodes and nodes to communities, while *Build Network* does all the network wiring. Steps 7-9 are implemented in module "Transition".

1: $Community\ Size\ Sequence, Node\ Degree\ Sequence \leftarrow Sequences\ from\ User$
2: $Build\ Communities\ @\ T_n \leftarrow Community\ Size\ Sequence, Node\ Degree\ Sequence$
3: $Build\ Network @ T_n \leftarrow Communities$
4: **while** $Remaining\ TimeSteps \neq 0$ **do**
5: $\quad Community\ Size\ Sequence, Node\ Degree\ Sequence \leftarrow Sequences\ from\ User$
6: $\quad Build\ Communities\ @\ T_{n+1} \leftarrow Community\ Size\ Sequence, Node\ Degree\ Sequence$
7: $\quad Network\ @\ T_n \leftarrow user\ Events$
8: $\quad Flow\ Nodes\ from\ T_n\ to\ T_{n+1} \leftarrow Search\ Most\ Similar\ Transition$
9: $\quad Build\ Network\ @\ T_{n+1}$
10: $\quad Report\ Data\ for\ T_n\ to\ T_{n+1}$
11: $\quad Network\ @\ T_n \leftarrow Network\ @\ T_{n+1}$
12: $\quad TimeSteps \leftarrow TimeSteps - 1$
13: **end while**
14: $Output\ Temporal\ Network$

Syntgen outputs textual information as the network is created overtime, including network metrics, network events and other supporting information. Syntgen also produces the full temporal network in machine readable format that can be input directly to the Gephi [4] visualization tool.

3.2 *Creating a static network*

Creating a T_0 static network involves the following steps:

- Receiving community size and node degree sequences from the user .
- Testing for "graphability" and requesting from the user new sequences if they are determined not to be "graphable".
- Randomly assigning nodes without substitution to communities from the bijection of intra (E) and total degrees (D) with $e \in E : e < |c|$.
- Wiring nodes using a modified version of the configuration model both for intra links as well as inter links respecting assortative specifications.

3.2.1 *Community, node sequences* Syntgen does not impose specific restrictions on the user input sequences beyond a coherent total number of nodes, and node intra community degrees that are less or equal to their respective total degree. It follows that Syntgen does not enforce community structure per se. The user must provide a ratio of intra to total degree that is conducive to community structure if a clustered network is preferred.

3.2.2 *Supplied distribution samplers* The user may opt to generate community and node sequences resorting to functionality provided by Syntgen. There are IID samplers of uniform, exponential and power law distributions. All of our supplied samples of sequence generators accept a ratio (r) of intra to total degree similar to the *mixing parameter* in the LFR benchmark [21]. To alleviate rounding artifacts that are more pronounced for nodes with low degree, we employ stochastic rounding instead of rounding to the nearest integer. The authors in [21] point out that allowing the ratio to change can lead to communities containing nodes that have a higher inter than intra degree (due to random fluctuations), but depending on usage, having a fixed intra to total degree ratio may be too restrictive on the desirable network topologies. Therefore, we let the user choose between a fixed $0 \leqslant r \leqslant 1$ or Bernoulli distributed ratio with $\mathbb{P} = r$.

Although we do not challenge if specifications as provided by the user or generated by the supplied distribution samplers are conducive to community structure, we do test for disconnected components inside communities by computing the algebraic multiplicity of the zero eigenvalue for the Laplacian of the adjacency matrix of the community. If higher than one, we warn the user, giving the option to continue or abort the network generation.

3.2.3 *Testing for graphability* To test user specifications for graphability as a simple network, we make use of the Erdös-Gallai condition [11] that states that a degree sequence D is graphable if:

$$\sum_{i=1}^{|D|} d_i \in \{2n : n \in \mathbb{N}\} \ \wedge \ \sum_{i=1}^{k} d_i \leqslant k(k-1) + \sum_{i=k+1}^{|D|} \min(d_i, k) \ \forall \, (1 \leqslant k \leqslant |D|) \tag{3.1}$$

where d is degree and $|D|$ the total number of nodes. We apply 3.1 to every single community using only E, the nodes intra degrees. If completed successfully, we move on to test graphability of the inter degrees sequence F. For this we reduce the network to a multi-graph where each community becomes a single node and the multi-links are the aggregate inter community links of the base network. It is obvious that $max(F) \leqslant \sum_{i=1}^{|F|} d_i - max(F)$ is a necessary condition for graphability, as otherwise there would be not enough links to satisfy the requirements of the largest community inter degree. But it is also not hard to see that if the total number of inter links is even, the condition above is not only necessary but also sufficient, or formally:

$$\sum_{i=1}^{|F|} f_i \in \{2n : n \in \mathbb{N}\} \ \wedge \ max(F) \leqslant \sum_{i=1}^{|F|} f_i - max(F) \tag{3.2}$$

To see why, consider a reduced network with 3 nodes (communities), c_1, c_2, c_3 and their respective inter node degree aggregation f_1, f_2, f_3, with $f_1 \geqslant f_2 \geqslant f_3$. If $f_1 = f_2 + f_3$, the network is obviously graphable. If $f_1 < f_2 + f_3$ and if $f_1 \in \{2n : n \in \mathbb{N}\}$ then $(f_2 \in \{2n : n \in \mathbb{N}\} \wedge f_3 \in \{2n : n \in \mathbb{N}\}) \vee (f_2 \in \{2n+1 : n \in \mathbb{N}_0\} \wedge f_3 \in \{2n+1 : n \in \mathbb{N}_0\})$ but as $f_1 \geqslant f_2$ one can always distribute links from c_1 to c_2 and c_3 such that the remainder degrees to be satisfied are equal. If $f_1 \in \{2n+1 : n \in \mathbb{N}_0\}$ then $(f_2 \in 2n+1 : n \in \mathbb{N}_0 \vee f_3 \in 2n+1 : n \in \mathbb{N}_0)$ in which case after one link is added between c_1 and c_2 or c_3 we revert to the previous case.

The above is a proof for a 3 community clustering. To generalize the proof, let's consider the addition of a community to the reduced network resulting in the clustering $C = \{c_1, \cdots, c_4\}$, with node degree aggregation $D = \{f_1, \cdots, f_4\}$, and $f_i \geqslant f_{i-1} : 2 < \forall i \leqslant 4$. If we use links from f_1 to satisfy f_4, we get: $f_1 - f_4 \geqslant f_2 \vee f_1 - f_4 < f_2$. If $f_1 - f_4 \geqslant f_2$ we reduce to the previous proof as $f_1 - f_4 + f_2 + f_3 \in \{2n : n \in \mathbb{N}\}$ and $f_1 - f_4 \leqslant f_2 + f_3$ (remember that $f_3 \geqslant f_4$).

If $f_1 - f_4 < f_2$ then to reduce to the previous 3-community proof we should have $f_2 < f_3 + (f_1 - f_4)$. This is easy to prove by contradiction as $f_2 > f_3 + (f_1 - f_4)$ is impossible, given that $f_1 > f_2$ would force $f_3 - f_4 < 0$ which violates the problem statement. So by contradiction and induction we prove the condition for graphability of the inter links part of the network.

In conclusion, a network with $|V|$ nodes and $|C|$ communities with size sequence S, each with a bijection of intra and inter degree sequences respectively $E^{c_i}, F^{c_i}\ \forall i \in \{1,\cdots,|C|\}$ is graphable under the condition in equation 3.3.

$$\forall i \in \{1,\cdots,|C|\} : \sum_{j=1}^{s_i} e_j^{c_i} \in \{2n : n \in \mathbb{N}\} \wedge (\sum_{j=1}^{k} e_j^{c_i} \leqslant k(k-1) + \sum_{j=k+1}^{s_i} \min(e_i,k) \forall (1 \leqslant k \leqslant s_i) \wedge \sum_{i=1}^{|V|} f_i \in \{2n : n \in \mathbb{N}\} \wedge max(F) \leqslant \sum_{i=1}^{|V|} f_i - max(F) \quad (3.3)$$

3.2.4 *Node assignment* Syntgen assigns nodes to communities randomly at time step T_0 from the pool of available nodes, avoiding communities with cardinality smaller than the node intra degree. From T_1 onwards nodes keep their community membership except to honour new community size sequences. The process of minimizing membership changes is covered in section 3.3. The user can indirectly control node degree temporal correlation by influencing degree selection from the supplied total degree sequence thru the shape parameters of a beta distribution used to sample the ordered sequence. When $\alpha = \beta = 1$ it reverts back to the uniform distribution.

3.2.5 *Configuration model* In Syntgen we based the generation of networks with user specified degree distributions on a modified version of the configuration model (CM) [2]. We use this modified version to wire nodes inside communities (one community at a time, as if they were separate networks) and to create inter community links.

The CM can create a network based on arbitrary sequences D of node degrees. To this end, it expands the sequence into a list of $(\sum D)$ node "stubs" that are randomly paired, creating links. See figure 2 for an example.

For our purpose, the standard CM presents two difficulties. The first is that nothing prevents a stub

FIG. 2: **Wiring the configuration model.** Example of setting up a first link between node stubs for a network with $n = 10$ nodes, 15 links and degree sequence $D = \{4,4,3,3,4,3,3,2,2,2\}$. Stubs are randomly chosen and as long as $\sum_{i=1}^{n} D_i$ is even, the process always concludes, albeit with multi edges and self loops.

from linking back to another stub belonging to the same node, or linking the same nodes multiple times, both of which are incompatible with our aim of building a simple network with no self loops and no

multi-links. The second is that we want to provide the user with some capacity to control joint degree distribution, while the CM results in the following fixed joint distribution:

$$p_{ij} = \frac{k_i k_j}{S-1} \tag{3.4}$$

where k_n is the number of stubs of node n and S the total number of stubs = $\sum D$.

As the network grows, the probability of self-loops and multi-links decreases. This probability varies with the actual node degree distribution, but it is not unreasonable to disregard self-loops and multi-edges when building the network (see figure 3 for an example), considering however that, (1) stub pairing can fail before all stubs are assigned (that is, a node can have unlinked stubs with no candidate stubs remaining), and (2) that equation 3.4 is no longer representative of the degree joint distribution.
The first problem can be circumvented by selectively rewiring nodes randomly from a pool of candidate nodes (those that could satisfy the outstanding stubs but are otherwise taken elsewhere). As we test for graphability beforehand, this completes successfully.

The second problem is less relevant as we aim to generate networks with tunable joint degree distribution. We modified the CM so that instead of connecting stubs IID over a uniform distribution, we connect them IID over a beta distribution from the ordered node degree sequence. As the probability density function (pdf) increases towards the rightmost side of the distribution domain, correlation increases, and vice-versa. The α and β shape parameters of the Beta distribution are specified by the user and enable flexible pdf shapes. Using these parameters, the user can influence the level of the network correlation, subject to structural cutoffs [6, 24].

3.3 *Minimizing Shared Information Distance*

Once we have constructed the network at time t, injected user changes, created the network N^{t+1} (all based on user specifications), and created adjustment communities for dead and new nodes, so that the number of nodes across steps remains the same, all that is left is to flow surviving nodes from one network to the next. We want to perform this node flow in such a way that the clusterings C^t and C^{t+1} are as similar as possible, in this way minimizing the changes beyond the user specifications. To measure the changes, we need a way of comparing clusterings. As mentioned previously, there are several approaches to this problem, from node pair counting to information based distance measures.

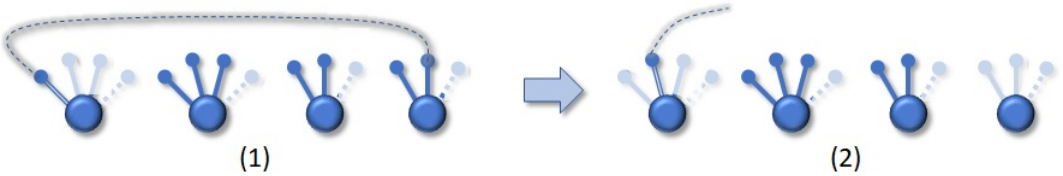
There is no best approach as explained in [23], and we have selected the metric therein proposed, the variation of information (VI) (see figure 4) for its algorithmic simplicity and the fact that it is a true metric [19], respecting positivity, symmetry, and the triangle inequality.

But let's set briefly aside the proposed method of comparing clusterings and consider the search space of feasible node flows between N_t and N_{t+1}. One way of looking at the problem is to coalesce N_t and N_{t+1} into the weighted bipartite network $G(C_t, C_{t+1}, U)$, where the nodes are the communities at successive timesteps and the U the weighted links representing the node flows between them.

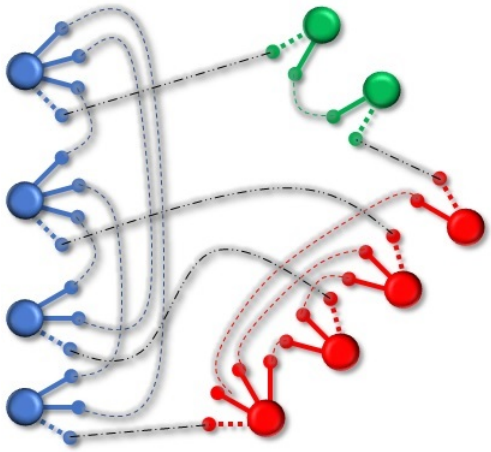
It is easy to see that the search space consists of the solutions to an under-determined system of Diophantine equations $Ax = B$ where A is the incidence matrix of the fully connected bipartite network $G = (C^t, C^{t+1}, L)$ and B is the vector $\{|C_i^t|_{i=1}^k \cup |C_j^{t+1}|_{j=1}^l\}$. As $rank(A) = |B| - 1$, one line of matrix A and the corresponding entry of vector B can be removed. Dimensionality can be further reduced as $sum[x]$ is known, and thus one element of x can be determined from the others. Every solution in the solution space is a vector x whose elements are the number of nodes (u) that should be transferred from communities in C^t to communities in C^{t+1}.

Formally we want to find C^{t+1} $s.t.$ $min(VI(C^t, C^{t+1}))$ with $Ax = B$ as defined above.

(a) We use a modified CM, and apply it to one community at a time for intra-community links (full stroke linestyle) and to the whole network for inter community links (dashed linestyle)



(b) Our modified version of the CM deletes from the link candidate list all the remaining stubs of the linking node (double stroke linestyle) (1), and all the remaining stubs of the linked node for subsequent stubs from the same node (2)



(c) Example of the fully connected network

FIG. 3: **CM Plots of a network** 3 communities (Blue, red and green), with community size sequence $\{4,4,2\}$ and total and intra degree sequences $\{4,4,4,3,3,3,3,2,2,2\},\{3,3,3,2,2,2,2,1,1,1\}$.

$$VI(X;Y) = -\sum_{i=1}^{k}\sum_{j=1}^{l}[log(\frac{r_{ij}}{p_i}) + log(\frac{r_{ij}}{q_j})] \tag{3.5}$$

FIG. 4: Variation of Information $VI(X;Y)$ where $X = \{x_1,\cdots,x_k\}$ and $Y = \{y_1,\cdots,y_l\}$ are clusterings of a given set S, with $n = |S|$, and $r_{ij} = \frac{x_i \cap y_j}{n}$, $p_i = \frac{|x_i|}{n}$ and $q_j = \frac{|y_i|}{n}$

In topological terms, the space of the solution is a lattice contained in an $n-1$ dimensional polytope, where $n = |C^t| \times |C^{t+1}|$, bounded by $n-1$ positive halfspaces (as $x_i \geqslant 0 : \forall i$), and by $|C^t| + |C^{t+1}| - 1$

hyperplanes defined by the equations in $Ax = B$. The number of solutions is equal to the number of lattice points. Counting lattice points in such a polytope is not an easy task [22] and quickly becomes intractable. Barvinok proposed an algorithm for lattice point counting in [3] that has been implemented in systems like Latte [1], software that counts lattice points and performs integration inside convex polytopes. Some experiments we ran in Latte that illustrate the size of the problem can be seen in table 3.

Clustering @ T	Clustering @ T+1	Number of solutions
$\{20,16,12\}$	$\{24,13,11\}$	$6.46000E+03$
$\{16,16,16\}$	$\{16,16,16\}$	$1.17810E+04$
$\{13,11,10,10\}$	$\{14,12,9,9\}$	$7.80605E+06$
$\{1300,1100,1000,1000\}$	$\{1400,1200,900,900\}$	$1.58534E+24$
$\{13,11,10,10,9\}$	$\{14,12,9,9,9\}$	$1.09501E+11$
$\{1300,1100,1000,1000,900\}$	$\{1400,1200,900,900,900\}$	$3.18145E+41$

Table 3: Solution space, as reported by the count function of Latte, for 6 examples of clustering pairs. As can be seen, flowing even a small number of communities generates a search space that is for all purposes intractable.

The solutions that are of interest to us will have a high degree of sparsity as we are looking for similar clusterings and, intuitively (and experimentally), a high dispersion of nodes across communities will not be conducive to similarity. Higher sparsity solutions correspond to surface features of the polytope lattice, that is, in decreasing sparsity order: vertices, edges, ridges, cells, facets and so on, basically the $1,\cdots,n-1$ elements of an n-dimensional polytope. We based our heuristic on this intuition, limiting our search space to the hull of the polytope (see figure 5) This can reduce the space significantly depending on the polytope geometry.

To scan the space we find the nullspace of A, formally $ker(A) = \{x \in \mathbb{N}^n : Ax = 0\}$, where $n = |C_t| \times |C_{t+1}|$, and one solution x_i to $Ax = B$. By linearly combining x_i with $ker(A)$ we can span the set of solutions to $Ax = B$. Finding a single solution is trivial, all that is needed is to flow nodes from C_t to C_{t+1} until no more nodes are left in C_t. In fact, although the problem as stated is NP-Hard (it is not difficult to reduce it to the partition problem, a well documented NP-complete problem [18]) there are easy ways of finding good solutions with a low VI value.

We have implemented a pool of these simple algorithms, with polynomial complexity on the number of communities, that were experimentally best performers amongst themselves. We have built a pool of 5 simple one-pass greedy heuristics with an objective function related either to sparsity or similarity and experimentally verified that, although any one of the 5 may achieve best performance, one of them clearly outperforms the others as the number of communities increase (see figure 6).

In our space scan heuristic, we use these solutions (or the best of them) as starting points for our space search. This is accomplished by an anytime algorithm that greedily searches the solution polytope hull for the lowest VI avoiding previously visited solutions (see algorithm 2). To halt the algorithm, the user can specify thresholds for search restart after a certain number of failed improvement trials and a certain number of failed restarts.

Although searching the hull of the polytope vastly reduces the search space in most circumstances, as the network grows the probability of improving on the results from the pool of simple algorithms decreases. For very large networks the user may select to proceed with the best result from the pool and forego the heuristic search for the sake of expediency.

Algorithm 2 Anytime greedy algorithm with taboo

```
1: currBest ← min(Solution(SimpleAlgorithmPool))
2: bestVI ← VI(currBest)
3: globalTries ← 0
4: visited ← ∅
5: while globalTries ⩽ globalTriesThreshold do
6:    localTries ← 0
7:    globalTries ← globalTries + 1
8:    while localTries ⩽ localTriesThreshold do
9:       localTries ← localTries + 1
10:      localBest ← MAXFLOAT
11:      for all v = vector ∈ ker(A) do
12:         n ← ((na × v + currBest ∈ solutionSpace) ∧ ((n+1) × v + currBest ∉ solutionSpace)
13:         newSol ← n × v + currBest
14:         if newSol ∉ visited then
15:            if VI(newSol) ⩽ localBest then
16:               localBest ← VI(newSol)
17:               newSolLocal ← newSol
18:            end if
19:         end if
20:         n ← ((n × v + currBest ∈ solutionSpace) ∧ ((n−1) × v + currBest ∉ solutionSpace)
21:         newSol ← n × v + currBest
22:         if newSol ∉ visited then
23:            if VI(newSol) ⩽ localBest then
24:               localBest ← VI(newSol)
25:               newSolLocal ← newSol
26:            end if
27:         end if
28:      end for
29:      if localbest = MAXFLOAT then
30:         Break Global Tries (Dead end)
31:      else
32:         visited ← visited ∪ newSolLocal
33:         if VI(newSolLocal) ⩾ bestVI then
34:            localTries ← localTries + 1
35:         else
36:            bestVI ← VI(newSolLocal)
37:            currBest ← newSolLocal
38:            localTries ← globalTries ← 0
39:         end if
40:      end if
41:   end while
42: end while
```

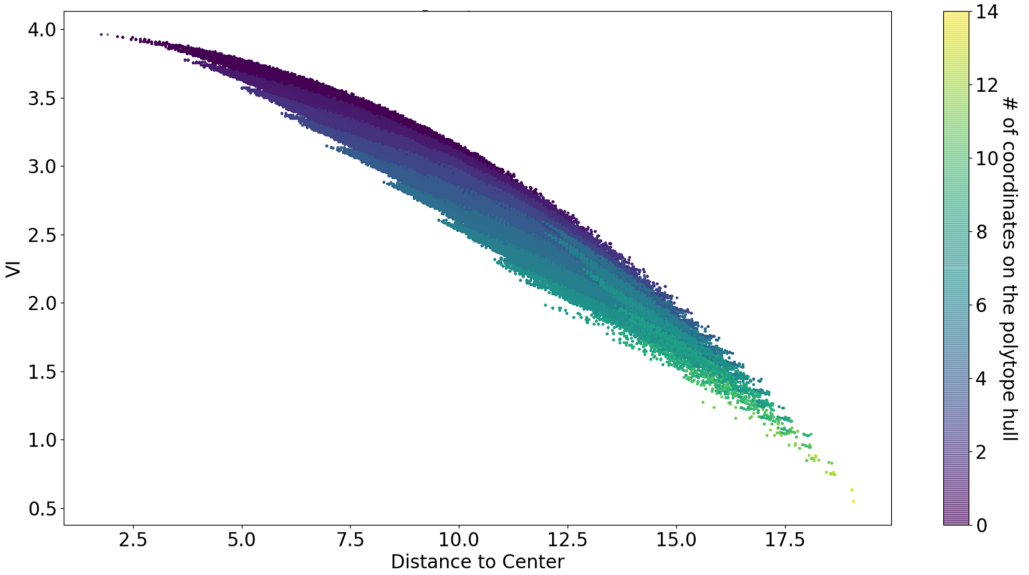


FIG. 5: **Comparing Clusterings similarity as a function of spacial location** Plot of all the 16,799,002 solutions of flowing a clustering with community size sequence of {13, 13, 12, 10} to {15,11,11,11}. We compare similarity, as measured by the variation of information, against distance to polytope center and number of polytope surface coordinates in the solution vector. The polytope "center" is computed as the vector $\left(\frac{max(x_i)\times n}{\sum_{j=1}^{f} max(x_j)}\right)_{i=1}^{f}$, where the vector x is the number of nodes flowing between communities (the sequence of edges weights of the fully connected bi-partite network), n is the total number of nodes, f the number of possible flows and $\max(x)$ the maximum number of nodes flowing between two communities. It is clearly visible that there is a strong positive correlation between these quantities.

In figure 7 an example of an exhaustive search of a very simple temporal network with a total of 279 solutions can be found to illustrate the method.

4. Experiments

Syntgen can generate many variants of temporal networks with community structure respecting user defined distributions of community sizes and node degree. In this section we present and discuss network metrics from generated networks according to varying input parameters. Given the time-slicing nature of Syntgen, most of the experiments highlight results from a single point in time, with the understanding that input parameters can be changed by the user on every time slice.

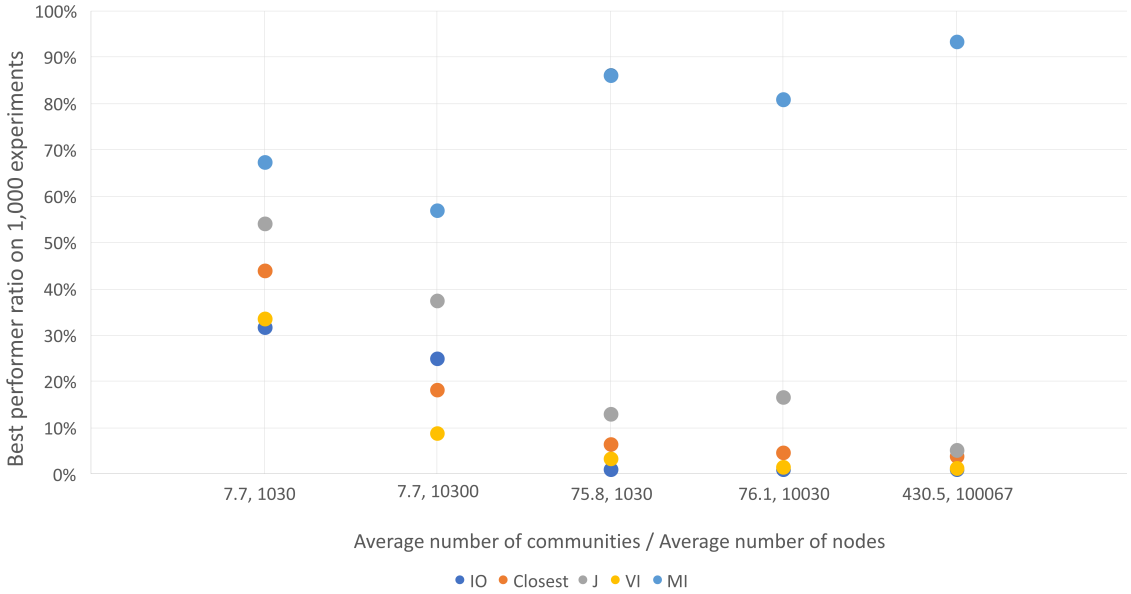


FIG. 6: **Relative performance of a pool of 5 simple algorithms to select a starting point for a space scan** All algorithms achieve top VI-based similarity in some of the 1,000 random runs, but one (MI, based on minimizing the increment of mutual information) vastly outperforms all others as the number of communities increases.

Sample Distributions

Syntgen provides distribution samplers of node degrees and community sizes. In figure 8 we can see an example of a power-law degree distribution and the effects of different rounding approaches.

Mix Ratio

We studied experimentally the impact of using a fixed vs Bernoulli distributed mix ratio (μ). As expected we did not observe significant differences between both approaches when run over 11 time steps, as can be seen on figure 9.

Joint Degree Assortativity

As discussed in section 3.2.5, joint degree is tunable by the user thru the shape parameters of the Beta distribution, affecting link generation. However, dependent on network structure, it may be impossible to generate a network with positive correlation. In figure 10 we plot all links of a network with 10,000 nodes and power-law distribution of node degrees and community size on a two-dimensional graph showing the degrees of their connected nodes. As the network is non directional the plots are symmetrical when reflected about the diagonal. The result of applying shape parameters to influence correlation can be clearly seen. We confirm previous findings, noting additionally that clustering has a

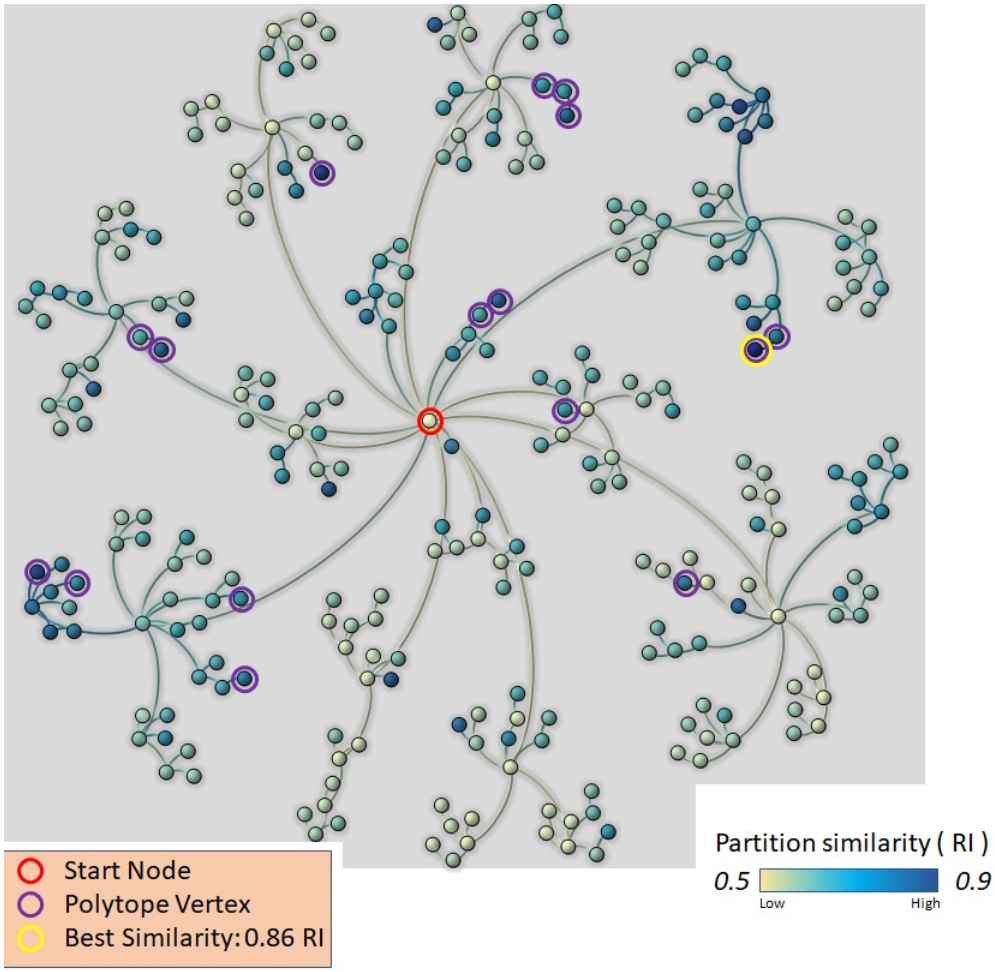
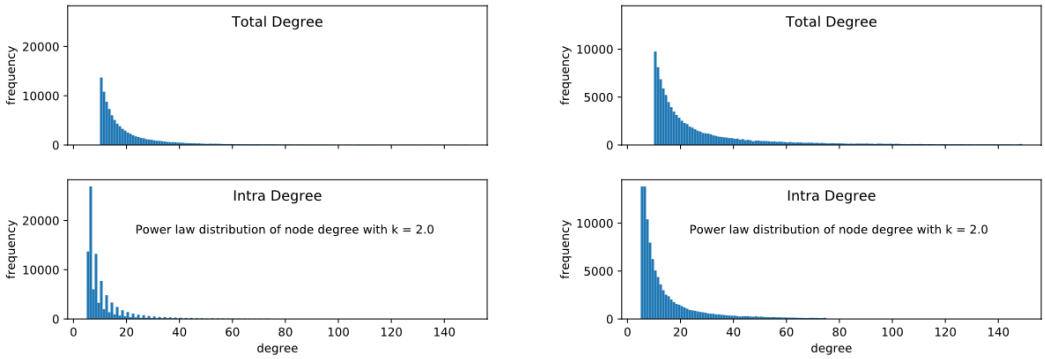


FIG. 7: **Example of a heuristic search to minimize information distance.** Example of heuristic search of a small network transition from $\{10, 8, 6\}$ to $\{12, 10, 2\}$. Note that vertex points of the solution lattice have higher than average similarity as measured by the Rand Index.

strong influence on correlation behaviour, potentially limiting the possibility of generating a positively degree correlated network.

Temporal Correlation

We use the same technique of sampling a beta distribution to influence the evolution of node degree. The user can change the distribution shape parameters to sustain a temporally homogeneous node degree, or to generate nodes that are cyclically active. Figure 11 shows the impact of varying shape parameters on the temporal node degree correlation.



(a) Rounding to nearest integer

(b) Stochastic rounding

FIG. 8: Node degree distribution of networks generated with 100,000 nodes, mix ratio of 0.7, and total node degree varying from 10 to 150. The artifact minimizing effect of stochastic rounding can clearly be seen in these examples.

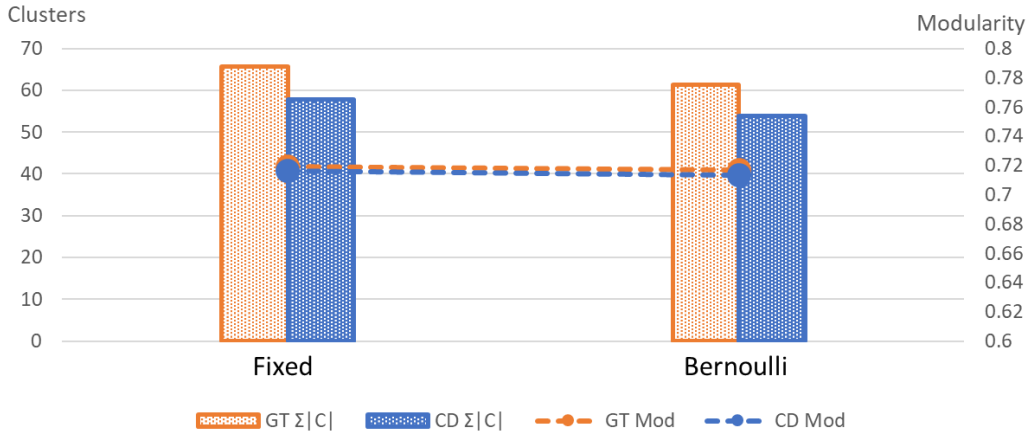
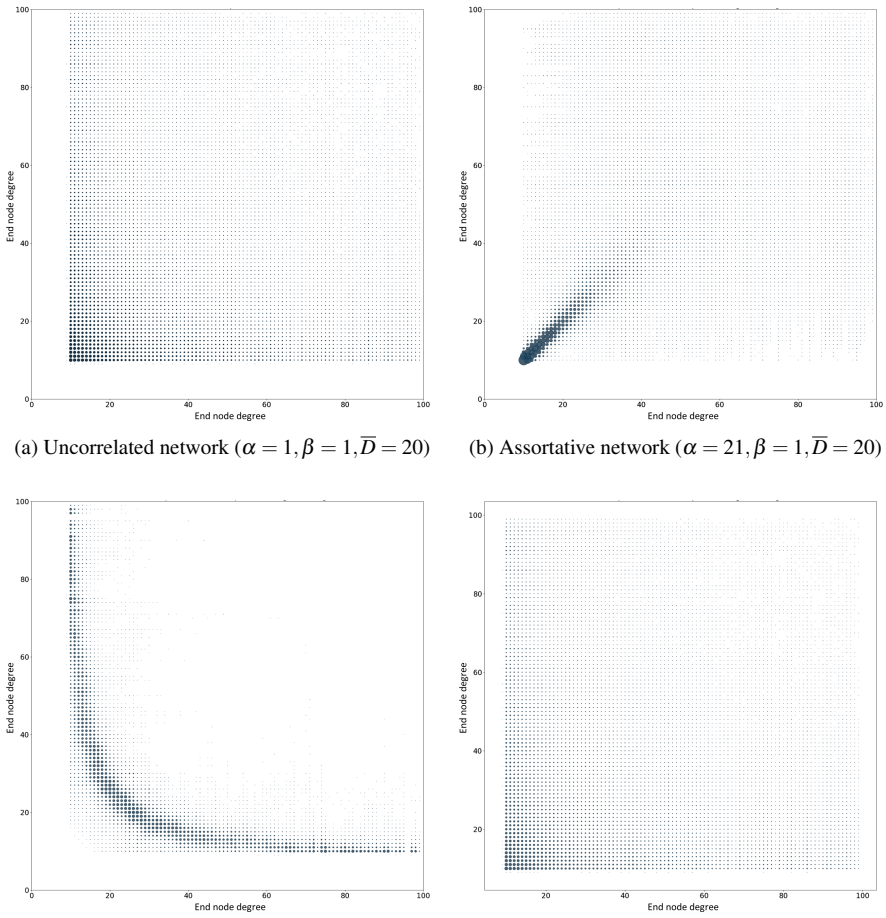


FIG. 9: **Fixed versus Bernoulli distribute mix ratio** Two networks averaged over 11 time steps, with 10,000 nodes, mix ratio $\mu = .7$, power law distribution of community size ($K_c = 1.5$) and node degree ($K_n = 2.5$), displaying ground truth modularity and modularity as computed by the community multilevel algorithm [5]. Differences in modularity between experiments are negligible. The differences in number of communities found between the ground truth and the community detection algorithm can be attributed to the resolution limits of the algorithm used for detection [13].

Sample Network

Currently Syntgen outputs machine readable networks in CSV format adequate for loading into Gephi [4]. In figure 12 an example of a Syntgen generated temporal network with lifecycle events can be seen.



(a) Uncorrelated network ($\alpha = 1, \beta = 1, \overline{D} = 20$)

(b) Assortative network ($\alpha = 21, \beta = 1, \overline{D} = 20$)

(c) Dissortative network ($\alpha = 1, \beta = 21, \overline{D} = 20$)

(d) Failed assortative network ($\alpha = 21, \beta = 1, \overline{D} = 25$)

FIG. 10: Assortative Experiments on power-law networks with 10,000 nodes, varying the average degree from 20 to 25 and maximum degree from 100 to 500, with an average community size of ≈ 168. Every point on the chart is a link with (x, y) coordinates representing the connecting nodes degrees. The point size is directly proportional to the total number of links with equal coordinates. As can be seen, as we increase the average node degree from 20 to 25 by increasing maximum node degree from 100 to 500, it is no longer possible to generate a correlated network with stated metrics even with aggressive beta distribution shape parameters.

A simple analysis of the transition from T_2 to T_3 can be found in table 4. The events were categorized as a function of the Jaccard Index [17] between communities, based on an external threshold to indicate community continuation.

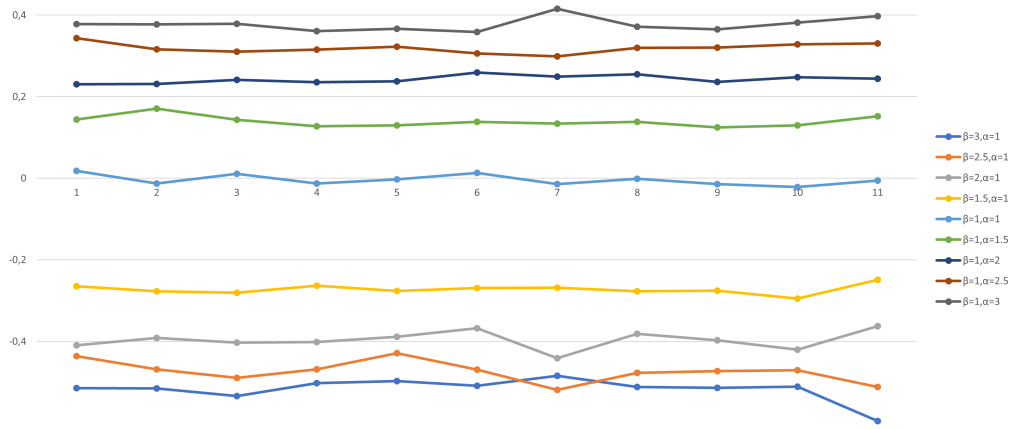


FIG. 11: **Temporal node degree correlation as a function of the Beta Distribution shape parameters** Evolution of the node degree Pearson's correlation at 11 successive time steps for different specifications of the Beta distribution shape parameters.

5. Remarks, Discussion and Conclusion

Syntgen is a random network generator with constraints. Links between nodes are created independent and identically distributed (IID) over explicit and implicit user specifications. Although a single instance may deviate significantly from the required specifications, the average of a set of generations will converge asymptotically to those specifications. The size of the network also has an impact on how closely specifications can be followed. For example, a network with a low average number of communities, will have community size distributions that are likely not recognizable when compared to parameter derived expectations.

It is also of note that, although Syntgen proceeds basically IID when wiring the network, every time a "dead-end" is encountered on graphable specifications, the affected process is restarted after rewiring adjustments. For instance, if, while generating intra links inside a community, a node exhausts its list of candidate nodes before satisfying its degree, other established links will be broken so that the process can proceed to satisfaction. This, in practice, "breaks" the IID aspect of the system, although for most specifications the impact will be marginal depending on the density of the communities and of the network.

Joint degree distribution specifications and node affinity over time is influenced by user parameters, but is also restricted by network wiring requirements and structural cutoffs [10]. This is the reason why it is not possible to directly specify node joint degrees or temporal correlation.

The ratio of intra to total node degree has a direct impact on the clustering modularity at any given time. The only node information kept by Syntgen is the intra-community and inter-community node degree that the user provides, and its linked nodes and community membership, generated by Syntgen. If specifications are not conducive to a clustered network, community membership will not be recovered from the network structure.

In fact, modularity is affected not only by the above ratio, but also by assortativity specifications. In a highly correlated network it is possible that the clustering generated by Syntgen does not exhibit maximum modularity, which can be verified experimentally. The intuition is that, as nodes exhibit connection preferences, communities within communities may appear, resulting in improved modularity

Community	Event @ end of time T_2
3	Continues shrinking in 3
10	Continues shrinking in 10
2	Continues shrinking in 2
6	Continues growing in 6
9	Continues growing in 9
8	Split into [12, 11]
8	Merged Into 11
1	Continues shrinking in 1
4	Continues growing in 4
5	Continues in 5
7	Merged Into 11
Community	Event @ beginning of time T_3
12	from Split 8
2	Continued shrinking from 2
3	Continued shrinking from 2
10	Continued shrinking from 10
9	Continued growing from 9
6	Continued growing from 8
1	Continued shrinking from 1
4	Continued growing from 4
5	Continued from 5
11	from Split 8
11	Merged from [8, 7]

Table 4: **Community events on time step transition**
Note community 8 as it splits into 12 and merges into 11 continuing in both communities

with a larger number of communities.

The main aim of Syntgen is to provide researchers in network science a tool flexible enough to generate temporal networks that approximates the topology observed in empirical networks. Syntgen can help where real data is not easily accessible, but whose structure and topology is known. In the process of building Syntgen, we developed a method to determine the graphability of intra and inter node degree and community size sequences, and a heuristic to find the node flow that results in the closest clusterings at successive time steps, given a network and a community size sequence.

We plan to use the developed search heuristic to determine change points in temporal networks with community structure. The intuition is that change points are correlated with a peak in community activity which would be detected as an increase in the dissimilarity gap between successive snapshots of the network. The gap to the (near) optimal flow would be a proxy of intensive change.

Other extensions to our work include the usage of Syntgen to evaluate community detection algorithms on temporal networks and analysing syntgen capabilities to reproduce empirical systems.

Acknowledgment and Funding

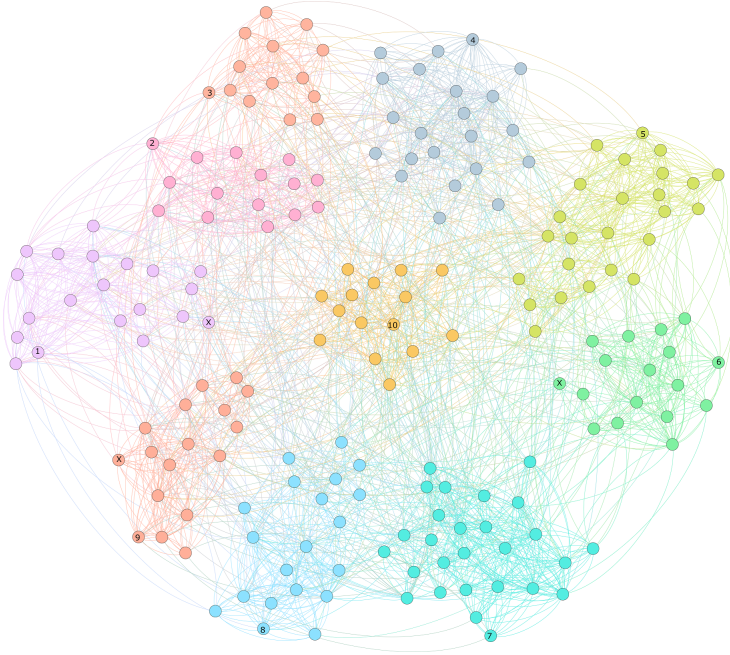
This project was partially supported by FCT (Fundação para a Ciência e Tecnologia) through project UID/Multi/04466/2019. The authors are thankful to Gergeley Palla and his team for relevant topical

discussions and to Maria J. Pereira for careful review of the draft manuscript.

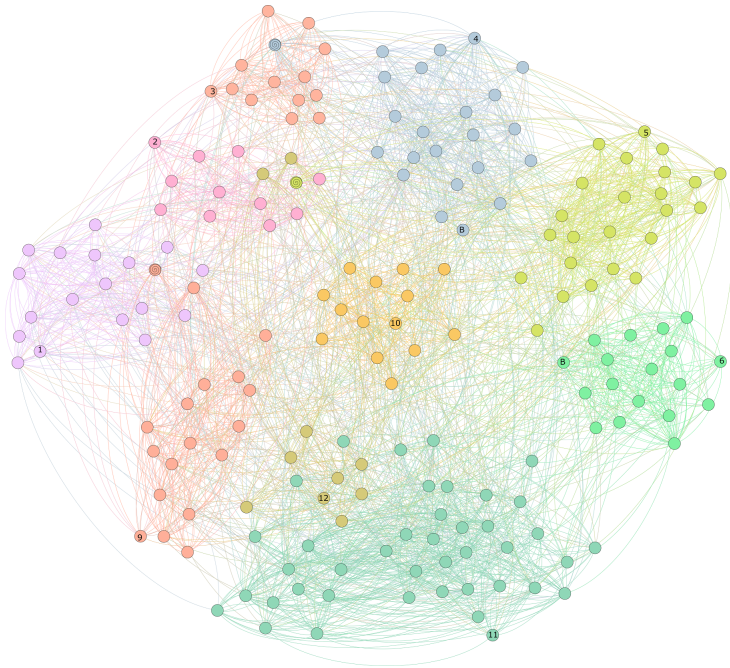
REFERENCES

1. BALDONI, V., BERLINE, N., DE LOERA, J. A., DUTRA, B., KOPPE, M., MOREINIS, S., PINTO, G., VERGNE, M. & WU, J. (2014) A User 's Guide for LattE integrale v1.7.2. .
2. BARABÁSI, A.-L. (2015) NETWORK SCIENCE. 4. The scale-free property. *Network Science*, pp. 1–57.
3. BARVINOK, A. & POMMERSHEIM, J. E. (1999) An Algorithmic Theory of Lattice Points in Polyhedra. **38**, 91–147.
4. BASTIAN, M., HEYMANN, S. & JACOMY, M. (2009) Gephi: An Open Source Software for Exploring and Manipulating Networks. *Third International AAAI Conference on Weblogs and Social Media*, pp. 361–362.
5. BLONDEL, V. D., GUILLAUME, J.-L. & LEFEBVRE, E. (2008) Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*.
6. BOGUNA, M., PASTOR-SATORRAS, R. & VESPIGNANI, A. (2003) Cut-offs and finite size effects in scale-free networks. (1), 1–5.
7. CHOUDUM, S. A. (1986) A simple proof of the Erdos-Gallai theorem on graph sequences. *Bulletin of the Australian Mathematical Society*, **33**(1), 67–70.
8. CLAUSET, A., SHALIZI, C. R. & NEWMAN, M. E. J. (2009) Power-law distributions in empirical data.. *SIAM Review*, **51**(4), 661–703.
9. CLAUSET, A. S. F. I. (2013) The configuration model. *Network Analysis and Modeling, CSI 5352, Lecture 11*, (October), 1–6.
10. DOROGOVTSEV, S. N. & MENDES, J. F. (2002) Evolution of networks. *Advances in Physics*, **51**(4), 1079–1187.
11. ERDÖS, P. & GALLAI, T. (1960) Gráfok előírt fokú pontokkal. *Matematikai Lapok*, **11**, 264–274.
12. EULER, L. (1736) Solutio problematis ad geometrian situs pertinentis. *Comentarii academiae scientarum Petropolitanae*, **8**, 128–140.
13. FORTUNATO, S. & BARTHÉLEMY, M. (2007) Resolution limit in community detection.. *Pnas*, **104**(1), 36–41.
14. FORTUNATO, S. & HRIC, D. (2016) Community detection in networks: A user guide. *Physics Reports*, **659**, 1–44.
15. GRANELL, C., DARST, R. K., ARENAS, A., FORTUNATO, S. & G??MEZ, S. (2015) Benchmark model to assess community structure in evolving networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, **92**(1), 1–11.
16. GREENE, D. (2010) Tracking the Evolution of Communities in Dynamic Social Networks. *2010 International Conference on Advances in Social Network Analysis and Mining : ASONAM 2010 : proceedings 2010*.
17. JACCARD, P. (1912) THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1. *New Phytologist*, **11**(2), 37–50.
18. KORF, R. E. (1998) A complete anytime algorithm for number partitioning. *Artificial Intelligence*, **106**(2), 181–203.
19. KRASKOV, A., STÖGBAUER, H., ANDRZEJAK, R. G. & GRASSBERGER, P. (2003) Hierarchical Clustering Based on Mutual Information. *arXiv:q-bio/0311039v2*.
20. LANCICHINETTI, A. & FORTUNATO, S. (2009) Community detection algorithms: A comparative analysis. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, **80**(5), 1–12.
21. LANCICHINETTI, A., FORTUNATO, S. & FILIPPO RADICCHI (2008) Benchmark graphs for testing community detection algorithms. *Physical Review E*.
22. LOERA, J. A. D. (2005) The Many Aspects of Counting Lattice Points in Polytopes. *Mathematische Semesterberichte*, **52**(2), 175–195.
23. MEILĂ, M. (2007) Comparing clusterings-an information based distance. *Journal of Multivariate Analysis*, **98**(5), 873–895.
24. NEWMAN, M. E. (2003) Mixing patterns in networks. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, **67**(2), 13.

25. NEWMAN, M. E. J. (2004) Algorithms for graph partitioning: A survey. *Social Networks*, **6**(2), 1–34.
26. NEWMAN, M. E. J., STROGATZ, S. H. & WATTS, D. J. (2000) Random graphs with arbitrary degree distributions and their applications. .
27. PALLA, G., BARABÁSI, A.-L. & VICSEK, T. (2007) Quantifying social group evolution. *Nature*, **446**(7136), 664–667.
28. ROSSETTI, G. (2017) RD YN : graph benchmark handling community dynamics. *Journal of Complex Networks*, (January), 893–912.
29. STANTON, I. & PINAR, A. (2011) Prescribed Joint Degree Distribution. *CoRR*, **abs/1103.4**, 1–29.
30. TRIPATHI, A., VENUGOPALAN, S. & WEST, D. B. (2010) A short constructive proof of the Erdo{double acute}s-Gallai characterization of graphic lists. *Discrete Mathematics*, **310**(4), 843–844.
31. WAGNER, S. & WAGNER, D. (2007) Comparing Clusterings - An Overview. *KITopen*, **4769**(001907), 1–19.
32. YANG, Z., ALGESHEIMER, R. & TESSONE, C. J. (2016) A Comparative Analysis of Community Detection Algorithms on Artificial Networks. *Scientific Reports*, **6**(August), 30750.



(a) Network at time T_2 with 10 communities, numerically and color identified.



(b) Network at time T_3 with 9 communities with merge and split events.

FIG. 12: Nodes marked "X" are examples of nodes to be killed across the transition to T_3. Nodes with a spiral are example of nodes that changed communities, and nodes marked "B" are examples of new born nodes. Community 8 split from T_2 to T_3 into community 12 and split-merged with community 5 to community 11.