SIAM J. MATH. DATA SCI.
Vol. 2, No. 2, pp. 505–528

Network Interpolation[*]

Thomas Reeves[†], Anil Damle[‡], and Austin R. Benson[‡]

Abstract. Given a set of snapshots from a temporal network we develop, analyze, and experimentally validate a so-called network interpolation scheme. Our method allows us to build a plausible, albeit random, sequence of graphs that transition between any two given graphs. Importantly, our model is well characterized by a Markov chain, and we leverage this representation to analytically estimate the hitting time (to a predefined distance to the target graph) and long-term behavior of our model. These observations also serve to provide interpretation and justification for a rate parameter in our model. Finally, through a mix of synthetic and real-world data experiments we demonstrate that our model builds reasonable graph trajectories between snapshots, as measured through various graph statistics. In these experiments, we find that our interpolation scheme compares favorably to common network growth models, such as preferential attachment and triadic closure.

Key words. network science, interpolation, dynamic networks

AMS subject classifications. 90B15, 05C82, 68R10

DOI. 10.1137/19M1268380

1. Introduction.

Dynamic networks for temporal interactions of complex systems are a pervasive model throughout the sciences [21]. They are used to analyze, for example, interactions in social networks [13, 30], communication systems [27, 29], digital currency transactions [26, 40], and protein-protein interactions [18, 25]. Often, these dynamic datasets are recorded as a sequence of "snapshots" (also called "slices" [36] or "layers" [24]), where the snapshot represents the network at a single point in time or an aggregation of data over a period of time. A sequence of snapshots is often the fundamental type of data used to derive methods for dynamic network analysis [3, 11, 17, 19].

Even if dynamic interactions occur in real time, there are a number of reasons why one may only have access to a sequence of snapshots. A principal reason is that data may only be collected at regular intervals. Specifically, such scenarios are common in survey data. For example, sociological studies record social networks of groups at different points in time [9, 14, 34], and U.S. Census data, such as the Survey of Income and Program Participation, record job movement by surveying households at regular intervals. An online analogue to offline surveys is Web crawling. In this scenario, sequences of snapshot networks are recorded from a sequence of crawls that collect network data and are subsequently analyzed for their dynamic structure [4, 31, 35]. In other cases, temporal network data are aggregated upon

[*] Received by the editors July 1, 2019; accepted for publication (in revised form) March 23, 2020; published electronically June 16, 2020.
https://doi.org/10.1137/19M1268380

Funding: This research was supported by NSF Award DMS-1830274, ARO Award W911NF19-1-0057, and ARO MURI.

[†] Center for Applied Mathematics, Cornell University, Ithaca, NY 14853 (tr352@cornell.edu).

[‡] Department of Computer Science, Cornell University, Ithaca, NY 14853 (damle@cornell.edu, arb@cs.cornell.edu).

public release. This happens due to privacy concerns in biomedical data [5, 7] or because the interaction is associated with a regularly scheduled event, such as coauthorship of a computer science conference paper in a given year [30, 32].

Independent of the manner in which a sequence of snapshots is obtained, a natural problem is inferring what happens in the network between the snapshots, when the underlying true data are not available. Such reverse engineering would enable better real-time data analysis, localization of structural changes in the graph, and understanding of social, financial, or biological dynamics. More generally, we would like to generate a plausible sequence of graphs that takes us from one snapshot to the next. Beyond enabling exploration of the network between snapshots, such a process is valuable in the development of synthetic data for streaming algorithms by providing test sets (sequences of graphs) anchored to the real data.

A natural first approach to this problem is to appeal to network growth models, such as preferential attachment [1, 28], triadic closure [22, 23], mixture models [38], or stochastic actor-oriented models [43]. In this setup, one would start with a snapshot and use a network growth model, perhaps with parameters determined by the structure of the next snapshot, as a guess at how the network evolves. However, there is no guarantee that after an appropriate number of steps, such growth models will be close in structure to the next snapshot, and indeed we see that this is the case in our experiments outlined in section 5. Across all of our experiments, the behavior we observe with these models is that when initiated with one snapshot, they do not have the same structure of the network at the time of the next snapshot.

Colloquially, employing a growth model looks like using an extrapolation method to try to move from one snapshot to the next—the only considerations are the choice of the parameters in a growth model and the starting snapshot. In contrast, because we have a sequence of snapshots, a more apt analogy is to try to build an interpolation strategy. Here we will use both the starting and ending snapshots and explicitly generate a path between them. To illustrate the subsequent discussion, Figure 1 shows four graphs produced in an instance of our network interpolation model.

We propose a temporal network model that, given two sequential snapshots, provides a path of *graphs* that begins at one snapshot and ends exactly at the next snapshot—i.e., it exactly interpolates the snapshots. More specifically, the model provides a feasible sequence of additions and deletions of edges that transitions between the two snapshots. We stress that a key benefit of our model is that it does *not* simply interpolate one-dimensional statistics of the snapshots (such as the clustering coefficient). It actually provides a sequence of graphs whose statistics interpolate those of the snapshots.

Of course, absent sufficient additional side information, there could be an infinite number of potential sequences from one graph to the next. Thus, our model is determined by the process through which we build the interpolation. More specifically, our model is based on a Markov chain on graphs. Given a starting graph G and a target graph H, our model makes a sequence of edits to the graph G; at each step, a biased coin determines whether we increase or decrease the edit distance between H and G by one, where the bias depends on the current edit distance.

While our model is a Markov chain on the space of all graphs of appropriate size, we show how to analyze its structure using a much simpler Markov chain on the set of edit distances between the starting and target graphs. This reduces the dimension of the chain from exponential in the number of nodes to quadratic in the size of the starting and target graphs.

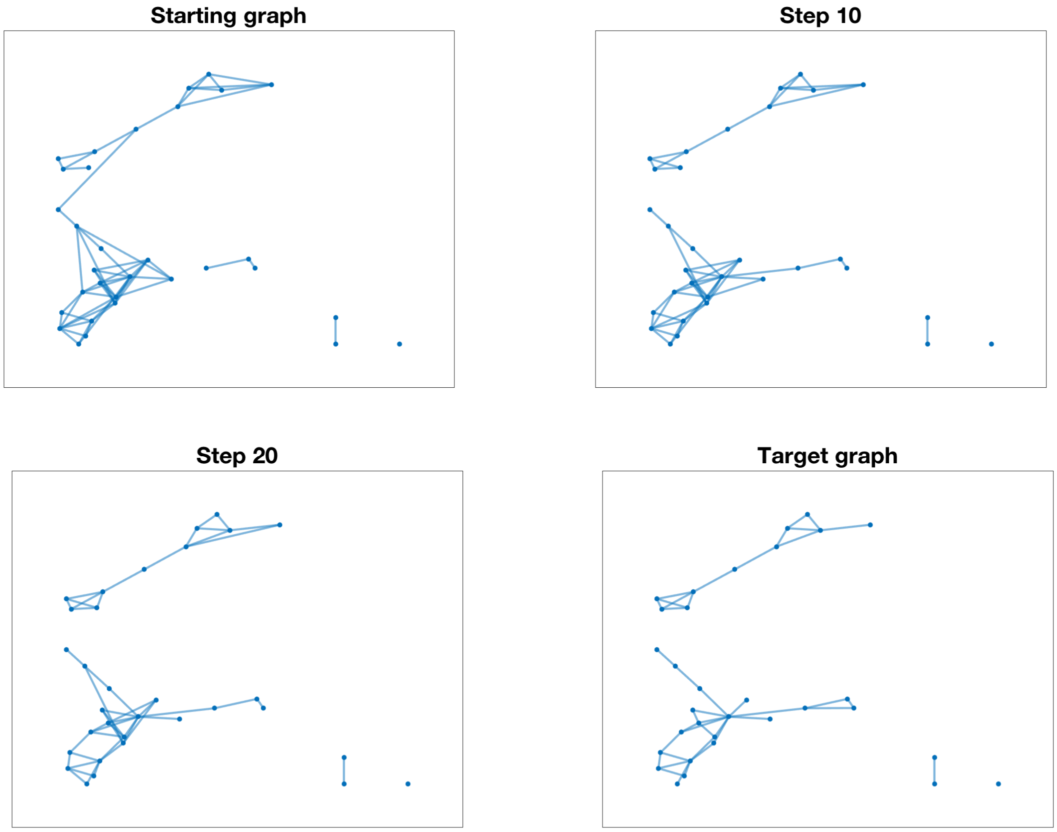


Figure 1. *Example graphs from an instance of our network interpolation model. The top left is the starting graph (Step* 0*), and the bottom right is the target graph (which is reached at Step* 31*); these two graphs are snapshots from a dynamic social network constructed from survey responses* [9]*. The other two graphs interpolate between the starting and target graphs and occur at the indicated intermediate steps indicated, where a single edge is added or deleted in each step (the target edit distance is set to* $d_t = 0$*, and the rate parameter is set to* $s = 1$*).*

By assuming a flexible bias parameter in the type of transition (increasing or decreasing edit distance), we are able to provide an analytic expression for the expected hitting time of the Markov chain, thereby characterizing the expected number of edge additions and deletions needed to reach the target graph as a function of a rate parameter. We also theoretically show that the limiting distribution is concentrated around the target graph. Our theory is also validated through numerical experiments.

A further benefit of our model is that it may be adapted beyond just the interpolation setting. Specifically, rather than being asked to explicitly hit the target graph, it can also evolve to a graph within a given edit distance of the target graph and then fluctuate around that point. This is useful in several settings. For example, one may assume a given snapshot is noisy and matching it exactly may not be desirable. Alternatively, this process may be used to construct long-term trajectories of graphs that fluctuate around a given target. These trajectories, anchored by real-world graphs, may then be used in the development and testing of streaming algorithms.

Experimentally, we use our model to analyze a number of synthetic and real-world network snapshot sequences. We show that our model can interpolate between snapshots, while common extrapolatory growth models deviate substantially from the exhibited structure even when we fit growth parameters to the snapshots. One key example is that our network interpolation scheme often maintains clustering between snapshots (as measured by the mean and global clustering coefficients), whereas growth models (even ones based on triadic closure) fail to reasonably represent the clustering between snapshots in communication, social, and collaboration networks.

2. Related work.

The most relevant models in the literature are snapshot models [15, 44], which consist of probabilistically generated graphs whose parameters (such as edge connection probabilities and vertex labels) vary randomly over a series of discrete time steps. Typically, the number of snapshots is very small compared to the size of the entire graph, whereas the granularity of our method is at a per-edge level. The main goal of existing snapshot studies is statistical inference of the model parameters. For example, Bhattacharjee, Banerjee, and Michailidis [8] consider a series of independently drawn graphs with a given community structure that exhibits a sudden change and study the statistical estimation of the time of the change. We perform a similar change-point detection experiment in section 4.

Importantly, snapshot models do not capture the fine-grain changes that take place in a dynamic network. Network data are becoming increasingly available through fine-grain measurements of temporal interactions [12, 21, 33, 41], and snapshot graphs are designed to capture coarse-grain structure. Also, snapshot models do not yet fully harness temporal structure to identify the emergence of structure in an *evolutionary* framework. Aldecoa and Marín [2] head toward this direction by proposing a rewiring process from one graph to another as a benchmark for evaluating community detection algorithms. In order to tackle the problem of understanding structure, we need fine-grain models for network evolution.

Another relevant model is the stochastic actor-oriented model [43]. Nodes are conceptualized as actors who control their ties to the other actors in the network. Each actor can change ties at time points determined by a Poisson rate function, and the probability of the actor toggling a tie given snapshots is estimated [9]. By contrast, our model processes edges one by one, does not characterize the individual nodes in the network, and operates on a global level.

Finally, another related idea to our research is so-called network archaeology, which reconstructs network histories given a present-day snapshot [37, 45]. These methods infer growth model parameters by running growth models in reverse to find likely ancestors of a graph. Our model instead interpolates from one snapshot to the next.

3. Network interpolation model.

We now concretely present and analyze our model for network interpolation based on given starting and target graphs. All code and data used for this paper are available at https://github.com/tr-maker/networkinterpolation.

3.1. Model description.

We initialize our model with the starting graph G and target graph H. For example, we could be given two snapshots of a social network and want to create a sequence of graphs interpolating between them. At each step, our model makes an update to G so that it represents the current graph. (The graph H remains fixed throughout the interpolation.) Our graph evolution model is in terms of the *graph edit distance* d between G

and H, which is the minimum number of *moves* (edge or vertex additions or deletions) needed to go from the former graph to the latter. Without loss of generality, we assume that G and H have the same number of vertices n,[1] and that moves consist only of edge additions and deletions. At each step, we change G so that

- with probability $\varphi(d)$, we make some *advancing* move that decreases the edit distance d by 1, or,
- with probability $1 - \varphi(d)$, we make some *regressing* move that increases the edit distance d by 1.

We have freedom in how the advancing and regressing moves are chosen.[2] Advancing moves (1) add an edge that is in H but not G or (2) delete an edge that is in G but not in H. Regressing moves (1) delete an edge in G that is also in H or (2) add an edge to G that is not in H. The simplest model to analyze is when all these possibilities are allowed, and each advancing (resp., regressing) move is chosen uniformly at random from the set of all possible advancing (resp., regressing) moves. However, we may, for example, wish to disallow adding an edge to G that is not in H. In such a model, we may have *outdated edges* (edges from the initial graph that are not in H) but never *false edges* (edges not in the initial graph or in H).

Another parameter we can control is how the advancing probability φ varies as a function of the current edit distance d to the target. Here we choose it as a sigmoid function of d that is centered at a given distance d_t. In the long term, this process generates graphs that fluctuate around graphs that are edit distance d_t from the target graph, and we call d_t the *target edit distance*. We also need to deal with boundary conditions of $d = 0$ and $d = d_m$, where $d_m = \binom{n}{2}$ is the maximum possible edit distance. More formally, φ is specified as follows:

- φ is a monotonically increasing function of d;
- $\varphi(0) = 0$, $\varphi(d_m) = 1$, and $\varphi(d_t) = \frac{1}{2}$.

These properties are realized if we define, for $0 < d < d_m$,

$$\varphi(d) = f\left(\frac{d - d_t}{s}\right), \tag{3.1}$$

where f is a sigmoid function and s is a rate parameter that controls the hitting time to d_t and the spread of distances around d_t in the time-averaged limiting distribution. The standard logistic function $f(x) = (1 + e^{-x})^{-1}$ makes our model analytically tractable, and we use this function throughout the paper.

Once we have chosen φ, Algorithm 3.1 compactly summarizes our model when we choose to run it for a fixed number of steps. (It is easily adaptable to settings where, e.g., we wish to terminate when the target distance is first reached—setting this target distance parameter to zero yields an interpolation scheme.) We note that if we want to store the dynamic graph at each step, we can save space by storing a sequence of edge updates to the initial graph. In addition, we do not need to store the set of regressing moves explicitly since they can be inferred from the advancing moves.

[1] This is not particularly limiting, as there is no restriction on G or H having multiple connected components or nodes with no edges.

[2] Because we are only modeling changes in the graph, we omit the possibility for the graph to remain the same. This removes any notion of time from our model, though it could be introduced by further modeling when the edits occur.

Algorithm 3.1 Network interpolation.

```
Input: $G$ = starting graph, $H$ = target graph, $T$ = number of steps, $s$ = rate, $d_t$ = target
    edit distance, $\varphi(\cdot, s, d_t)$ = advancing probability
initialize $\mathcal{X} = \{$advancing moves from $G$ to $H\}$
initialize $\mathcal{Y} = \{$regressing moves from $G$ to $H\}$
$d \leftarrow$ edit distance between $G$ and $H$
for $\tau = 1 : T$ do
    bool $\sim$ Bernoulli($\varphi(d)$)
    if bool then
        // Make an advancing move.
        sample a move from $\mathcal{X}$ and update $G$, $\mathcal{X}$, and $\mathcal{Y}$
        $d \leftarrow d - 1$
    else
        // Make a regressing move.
        sample a move from $\mathcal{Y}$ and update $G$, $\mathcal{X}$, and $\mathcal{Y}$
        $d \leftarrow d + 1$
    end if
end for
```

Algorithm 3.2 describes in more detail an efficient implementation of Algorithm 3.1 for undirected graphs given the adjacency matrices of the starting and target graphs. If A is the strictly upper triangular part of the adjacency matrix of the starting graph and B the strictly upper triangular part of the adjacency matrix of the target graph, then the matrix $U = B - A$ keeps track of the advancing and regressing moves.[3] An entry of 1 in U indicates an edge that is not in the current graph but is in the target graph, an entry of -1 in U indicates an edge that is in the current graph but is not in the target graph, and an entry of 0 in U indicates an edge that is both in the current graph and the target graph or not in either. Therefore, the number of nonzeros in U is the number of advancing moves (or the edit distance d between the current graph and the target graph) and the number of zeros in U is the number of regressing moves.

Our algorithm scales well with respect to space and time complexity. Before giving a more formal analysis, we first give a sense of the practical running time based on one of our larger experiments in section 5. There, we interpolate between 9 snapshots of a coauthorship graph with several hundred thousand nodes and edges in each snapshot. We set $s = 1$ as the rate parameter and $d_t = 0$ as the target edit distance, and we sequentially ran our algorithm by using consecutive snapshots as the respective starting and target snapshots until the target snapshot was reached. The entire sequence of 8 consecutive interpolations covered 2862784 steps and took 49.2 seconds in total, or 17.2 microseconds per interpolation step, on an early 2015 MacBook Pro with 2.7 GHz and 8 GB of RAM. (Our implementation is not highly optimized, and one possible approach for improving performance in practice would be to make the advancing and regressing moves in batches.)

[3] If instead the starting and target graphs are directed, then we let A be the entire adjacency matrix of the starting graph, B be the entire adjacency matrix of the target graph, and $U = B - A$.

Algorithm 3.2 Network interpolation (sketch of efficient implementation for undirected graphs).

```
Input: $A$ = strictly upper triangular part of adjacency matrix of starting undirected graph,
  $B$ = strictly upper triangular part of adjacency matrix of target undirected graph, $T$ =
  number of steps, $s$ = rate, $d_t$ = target edit distance, $\varphi(\cdot, s, d_t)$ = advancing probability
initialize $U = B - A$
$d \leftarrow$ number of nonzeros of $U$
for $\tau = 1 : T$ do
   bool $\sim$ Bernoulli($\varphi(d)$)
   if bool then
      // Make an advancing move.
      sample an index $i$ from the nonzero entries of $U$
      if $U(i) = 1$ then
         $A(i) \leftarrow 1$   // Add the corresponding edge to $A$.
      else
         $A(i) \leftarrow 0$   // $U(i) = -1$, so we delete the corresponding edge from $A$.
      end if
      $U(i) \leftarrow 0$
      $d \leftarrow d - 1$
   else
      // Make a regressing move.
      sample an index $i$ from the zero entries of $U$
      if $B(i) = 1$ then
         $A(i) \leftarrow 0$, $U(i) \leftarrow 1$   // Delete the corresponding edge from $A$.
      else
         $A(i) \leftarrow 1$, $U(i) \leftarrow -1$   // $B(i) = 0$, so we add the corresponding edge to $A$.
      end if
      $d \leftarrow d + 1$
   end if
end for
```

We now analyze the space and time complexity of our algorithm theoretically. If A and B are sparse, then U is sparse since it contains at most the combined number of nonzeros of A and B. As the algorithm progresses, U becomes sparser with each advancing move and denser with each regressing move. Due to our assumptions on the advancing probability $\varphi(\cdot, s, d_t)$, with high probability the sparsity of U will be on the order of the sparsity of A and B throughout the algorithm. In other words, the total storage required for the algorithm is on the order of the storage required to store the starting and target graphs.

As for the running time of the algorithm, initializing U takes time linear in the number of nonzeros of A and B. After that, each step of the algorithm depends on the complexity of the following three operations: (1) sampling a random index from the nonzero entries of U (for an advancing move), (2) sampling a random index from the zero entries of U (for a regressing move), and (3) updating an entry of U (and, similarly, updating an entry of A).

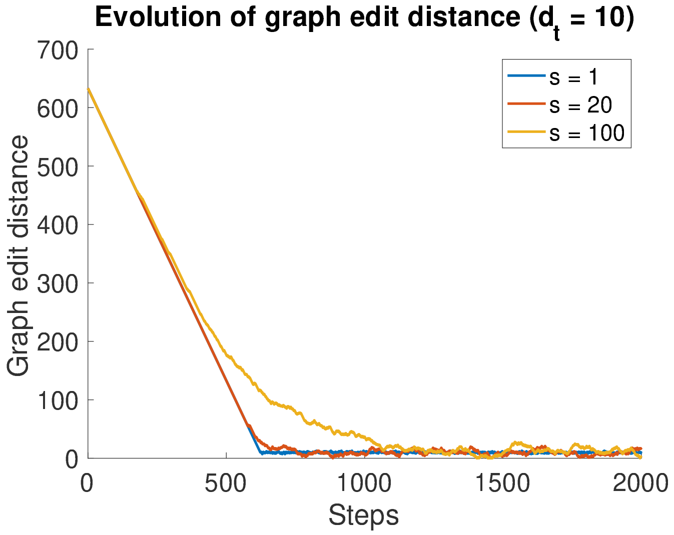


Figure 2. *Evolution of edit distance from an Erdős–Rényi random starting graph to a target graph of* 2 *communities, where transitions follow* (3.1). *The higher the rate parameter s, the longer the hitting time to $d_t = 10$ and the more spread out the distribution of edit distances.*

In the case of uniform sampling and sparse U, the three operations can be made to take amortized constant time. By storing the entries of U in a dynamically growing array and a hash map, where the entries of the array consist of the signed edges (1 or -1) in U and the hash map maps each signed edge to its index in the array, the operations of insertion, deletion, search, and sampling a uniformly random element can be performed in amortized constant time. In particular, this takes care of operations (1) and (3). To implement operation (2), we can use rejection sampling, drawing uniformly random edges from the set of all possible edges until we draw an edge whose corresponding entry in U is zero. If the number of nonzeros in U is linear in the number of vertices, then this process takes a constant number of iterations with high probability. Altogether, each step of the algorithm takes amortized constant time. Therefore, the total running time is on the order of the number of nonzeros of A and B plus the order of the number of steps T.

Figure 2 shows how various choices of s in (3.1) affect the series of edit distances from the target graph. For Figure 2 and the remainder of the figures analyzed in section 3.2, the starting graph is a 50-node Erdős–Rényi random graph with edge connection probability 0.5, the target graph is a 50-node stochastic block model divided into two equally sized clusters with in-cluster edge connection probability 0.9 and out-of-cluster edge connection probability 0.1, and the target edit distance d_t is equal to 10.

3.2. Model analysis. From Algorithm 3.1, our model is a Markov chain on graphs. However, many interesting properties of our model can be gleaned by looking at just the edit distances at each step of the interpolation. Our model naturally describes a stochastic process on edit distances $\{0, \ldots, d_m\}$; this process is in fact Markovian *provided that an advancing move and a regressing move are always possible for* $0 < d < d_m$. This very mild assumption holds exactly in the simplest case where all possible advancing and regressing moves are allowed,[4] and our theoretical estimates under this assumption closely match the experiments.

With the Markov assumption, $\varphi(i)$ describes the transition probability from edit distance i to edit distance $i-1$. The transition probability matrix therefore has $1-\varphi(i)$ above the diagonal and $\varphi(i)$ below the diagonal:

$$\begin{bmatrix} & 1 & & & & \\ \varphi(1) & & 1-\varphi(1) & & & \\ & \varphi(2) & & 1-\varphi(2) & & \\ & & \ddots & & \ddots & \\ & & & \varphi(d_m-1) & & 1-\varphi(d_m-1) \\ & & & & 1 & \end{bmatrix}. \tag{3.2}$$

With this, we analyze two important properties of our model: the limiting distribution of edit distances from the target graph (Proposition 3.1) and the hitting time to a distance d_t from the target graph (Proposition 3.2). For these two quantities, we give formulae that can be computed explicitly in terms of the rate parameter s. These propositions are therefore of practical importance for choosing the rate parameter in modeling. We then consider the properties of a random walk on the space of all graphs on n vertices (Proposition 3.3). For these theoretical results, we assume that φ is of the form in (3.1) with f the standard logistic function. To experimentally validate the propositions, we use the simplest version of the model, where each advancing (resp., regressing) move is chosen uniformly at random from the set of all possible advancing (resp., regressing) moves.

3.2.1. Time-averaged limiting distribution of edit distances. The Markov chain as described has period 2: The state at any given time step is either an even or an odd distance away from the initial edit distance. Nevertheless, the Markov chain has a time-averaged limiting distribution in the sense that for each state, the proportion of time spent in that state before step n converges as $n \to \infty$ [42].

The left eigenvector corresponding to the eigenvalue 1 of the transition probability matrix is represented in unnormalized form as follows:

$$\begin{bmatrix} \varphi(1)\cdots\varphi(d_m-1) \\ \varphi(2)\cdots\varphi(d_m-1) \\ (1-\varphi(1))\varphi(3)\cdots\varphi(d_m-1) \\ (1-\varphi(1))(1-\varphi(2))\varphi(4)\cdots\varphi(d_m-1) \\ \cdots \\ (1-\varphi(1))\cdots(1-\varphi(d_m-2)) \\ (1-\varphi(1))\cdots(1-\varphi(d_m-1)) \end{bmatrix}^T. \tag{3.3}$$

The time-averaged limiting distribution $v = \{v_i\}_{i=0}^{d_m}$ of the Markov chain is this eigenvector normalized by the sum of the entries. Although it is possible to do this eigenvector compu-

[4] It is not strictly true in the case that we disallow false edges (i.e., disallow adding an edge to the current graph G that is not in the target graph H) since every time an edge that is not in H is removed from G, the edit distance for which a regressing move is impossible decreases by one.

tation directly, we have the following simpler formula for approximately computing entries of the eigenvector around d_t.

Proposition 3.1. *Assume that* $d_m \geq 2d_t$. *There exists a constant* C *such that for every* $d_t \geq 2$ *and* $0 < \epsilon < 1$, *if* $s < C\frac{d_t^2}{\log 1/\epsilon}$, *then each component of the time-averaged limiting distribution satisfies the following approximation for* $0 < k < d_t$*:*

$$\left| v_{d_t \pm k} - \frac{e^{-\frac{k(k-1)}{2s}} + e^{-\frac{k(k+1)}{2s}}}{2\left(e^{-\frac{(d_t-1)d_t}{2s}} + 2\sum_{i=0}^{d_t-2} e^{-\frac{i(i+1)}{2s}}\right)} \right| < \epsilon. \tag{3.4}$$

Proof. We can write the entries of v recursively:

$$v_{i+1} = \frac{1-\varphi(i)}{\varphi(i+1)} v_i, \quad 0 \leq i < d_m. \tag{3.5}$$

From (3.5), we make two observations. First, if $\varphi(d_t + i) = 1 - \varphi(d_t - i)$ for all i, then

$$v_{d_t+i} = v_{d_t-i}. \tag{3.6}$$

Second, if we iterate (3.5) k times, then we get

$$v_{i+k} = \frac{1-\varphi(i)}{\varphi(i+k)} \left(\frac{1}{\varphi(i+1)} - 1\right) \cdots \left(\frac{1}{\varphi(i+k-1)} - 1\right) v_i,$$

so in particular

$$v_{d_t+i} = \frac{1/2}{f(\frac{i}{s})} \left(\frac{1}{f(\frac{1}{s})} - 1\right) \cdots \left(\frac{1}{f(\frac{i-1}{s})} - 1\right) v_{d_t}. \tag{3.7}$$

Now assuming that f is the standard logistic function, combining (3.6) and (3.7) gives that

$$v_{d_t \pm i} = \frac{1}{2}(1 + e^{-\frac{i}{s}})e^{-\frac{1}{s}} \cdots e^{-\frac{i-1}{s}} v_{d_t} = \frac{1}{2}(1 + e^{-\frac{i}{s}})e^{-\frac{i(i-1)}{2s}} v_{d_t} = \frac{e^{-\frac{i(i-1)}{2s}} + e^{-\frac{i(i+1)}{2s}}}{2} v_{d_t}. \tag{3.8}$$

If $d_m \geq 2d_t$, $d_t \geq 2$, and s is small enough relative to d_t, then we can approximate the time-averaged limiting distribution by assuming that v_i is supported on $0 < i < 2d_t$. Then (3.8) gives

$$1 = v_{d_t} + 2\sum_{i=1}^{d_t-1} \frac{e^{-\frac{i(i-1)}{2s}} + e^{-\frac{i(i+1)}{2s}}}{2} v_{d_t} + \eta v_{d_t}, \tag{3.9}$$

where the error

$$\eta < ce^{-\frac{d_t(d_t-1)}{2s}} \tag{3.10}$$

for some universal constant c. So

$$\frac{1}{v_{d_t}} = 1 + 2\left(\frac{1}{2} + \frac{1}{2}e^{-\frac{(d_t-1)d_t}{2s}} + \sum_{i=1}^{d_t-2} e^{-\frac{i(i+1)}{2s}}\right) + \eta = e^{-\frac{(d_t-1)d_t}{2s}} + 2\sum_{i=0}^{d_t-2} e^{-\frac{i(i+1)}{2s}} + \eta,$$

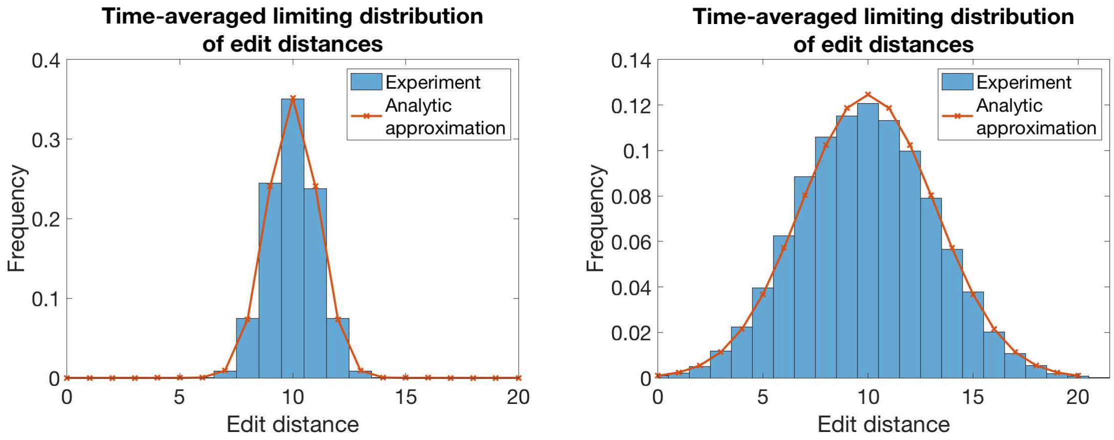


Figure 3. *Histogram of the time-averaged limiting distribution of edit distances of an evolution of a random graph to a target graph consisting of* 2 *communities, together with its analytic approximation (Proposition* 3.1*) for two rate parameters ($s = 1$, left; $s = 10$, right). The histograms are data from steps* 10001–20000 *of a simulation. The target distance $d_t = 10$.*

and so $v_{d_t} = \frac{1}{e^{-\frac{(d_t-1)d_t}{2s}}+2\sum_{i=0}^{d_t-2} e^{-\frac{i(i+1)}{2s}}+\eta}$. Going back to (3.8), we get

$$v_{d_t \pm k} = \frac{e^{-\frac{k(k-1)}{2s}} + e^{-\frac{k(k+1)}{2s}}}{2\left(e^{-\frac{(d_t-1)d_t}{2s}} + 2\sum_{i=0}^{d_t-2} e^{-\frac{i(i+1)}{2s}}\right) + 2\eta}. \tag{3.11}$$

The parenthesized term in the denominator is greater than 1, independent of d_t and s. So for (3.4) to hold, it suffices that η is at most some constant times ϵ, where the constant is also independent of d_t and s. The condition for s in the proposition follows from (3.10). ■

Figure 3 shows that the analytic formula reasonably approximates the limiting distribution of edit distances if the distribution of edit distances is not too spread out so that a symmetric approximation is reasonable.

3.2.2. Expected hitting time to the target edit distance.

The next proposition gives an analytic formula for the expected hitting time.

Proposition 3.2. *The expected hitting time h_{d_o} to the target edit distance d_t, starting at the initial edit distance d_o to the target graph, is*

$$h_{d_o} = (d_o - d_t) + 2\left(\sum_{k=1}^{d_m-d_o} e^{-\frac{k(k+1)}{2s}} \frac{1-e^{-\frac{k(d_o-d_t)}{s}}}{1-e^{-\frac{k}{s}}}\right). \tag{3.12}$$

In particular, the hitting time is monotonic in s.

Proof. Let h_i be the expected time to hit d_t starting from edit distance d_i. Then we have the linear recurrence

$$h_{d_t} = 0, \quad h_i = 1 + \varphi(i)h_{i-1} + (1-\varphi(i))h_{i+1}. \tag{3.13}$$

Let $d_o > d_t$ be the starting edit distance. We are interested in h_{d_o}. From (3.13), adding $\varphi(i)h_i - h_i$ to both sides, subtracting $\varphi(i)h_{i-1}$ from both sides, and dividing by $\varphi(i)$ gives

$$h_i - h_{i-1} = \frac{1}{\varphi(i)} + \left(\frac{1}{\varphi(i)} - 1\right)(h_{i+1} - h_i), \tag{3.14}$$

which is a recurrence in terms of the successive differences of hitting times. When $\varphi(d) = f\left(\frac{d-d_t}{s}\right)$ is the standard logistic function, the above equation simplifies to

$$h_i - h_{i-1} = 1 + e^{-\frac{i-d_t}{s}} + e^{-\frac{i-d_t}{s}}(h_{i+1} - h_i)$$

with initial condition $h_{d_m} - h_{d_m-1} = 1$. Rewriting, we get

$$h_k - h_{k-1} = 1 + 2\left(e^{-\frac{(d_m-1)-d_t}{s} - \ldots - \frac{k-d_t}{s}} + e^{-\frac{(d_m-2)-d_t}{s} - \ldots - \frac{k-d_t}{s}} + \cdots + e^{-\frac{k-d_t}{s}}\right). \tag{3.15}$$

Now sum (3.15) from $k = d_t + 1$ to d_m. The left-hand side is then the telescoping sum $h_{d_m} - h_{d_t} = h_{d_m}$, and we obtain

$$h_{d_m} = \sum_{k=d_t+1}^{d_m} \left(1 + 2\left(e^{-\frac{d_m-1-d_t}{s} - \ldots - \frac{k-d_t}{s}} + e^{-\frac{d_m-2-d_t}{s} - \ldots - \frac{k-d_t}{s}} + \cdots + e^{-\frac{k-d_t}{s}}\right)\right). \tag{3.16}$$

To find the value of h_{d_o}, we sum (3.15) from $k = d_o + 1$ to d_m. The left-hand side is the telescoping sum $h_{d_m} - h_{d_o}$. Plugging in (3.16) for the value of h_{d_m} gives

$$\begin{aligned}
h_{d_o} &= \sum_{k=d_t+1}^{d_m} \left(1 + 2\left(e^{-\frac{d_m-1-d_t}{s} - \ldots - \frac{k-d_t}{s}} + e^{-\frac{d_m-2-d_t}{s} - \ldots - \frac{k-d_t}{s}} + \cdots + e^{-\frac{k-d_t}{s}}\right)\right) \\
&\quad - \sum_{k=d_o+1}^{d_m} \left(1 + 2\left(e^{-\frac{d_m-1-d_t}{s} - \frac{d_m-2-d_t}{s} - \ldots - \frac{k-d_t}{s}} + e^{-\frac{d_m-2-d_t}{s} - \ldots - \frac{k-d_t}{s}} + \cdots + e^{-\frac{k-d_t}{s}}\right)\right) \\
&= \sum_{k=d_t+1}^{d_o} \left(1 + 2\left(e^{-\frac{d_m-1-d_t}{s} - \ldots - \frac{k-d_t}{s}} + e^{-\frac{d_m-2-d_t}{s} - \ldots - \frac{k-d_t}{s}} + \cdots + e^{-\frac{k-d_t}{s}}\right)\right) \\
&= (d_o - d_t) + 2\sum_{k=1}^{d_o-d_t} \left(e^{-\frac{k}{s}} + e^{-\frac{k}{s} - \frac{k+1}{s}} + \cdots + e^{-\frac{k}{s} - \frac{k+1}{s} \ldots - \frac{d_m-1-d_t}{s}}\right) \\
&= (d_o - d_t) + 2\left(e^{-\frac{1}{s}} \frac{1 - e^{-\frac{d_o-d_t}{s}}}{1 - e^{-\frac{1}{s}}} + e^{-\frac{3}{s}} \frac{1 - e^{-\frac{2(d_o-d_t)}{s}}}{1 - e^{-\frac{2}{s}}} + e^{-\frac{6}{s}} \frac{1 - e^{-\frac{3(d_o-d_t)}{s}}}{1 - e^{-\frac{3}{s}}} + \cdots\right). \quad ■
\end{aligned}$$

Figure 4 shows that the analytic expression for the hitting time is very close to the empirical average hitting time if enough terms are included in the sum in (3.12). This again shows that the Markov model approximates the actual dynamics of edit distances. For the analytic hitting times, we kept terms in the sum larger than machine precision; this gave 9 terms for rate $s = 1$ and 27 terms for rate $s = 10$. Figure 4 also shows how differently our model behaves for different s in terms of the spread of actual hitting times. Figure 5 quantitatively

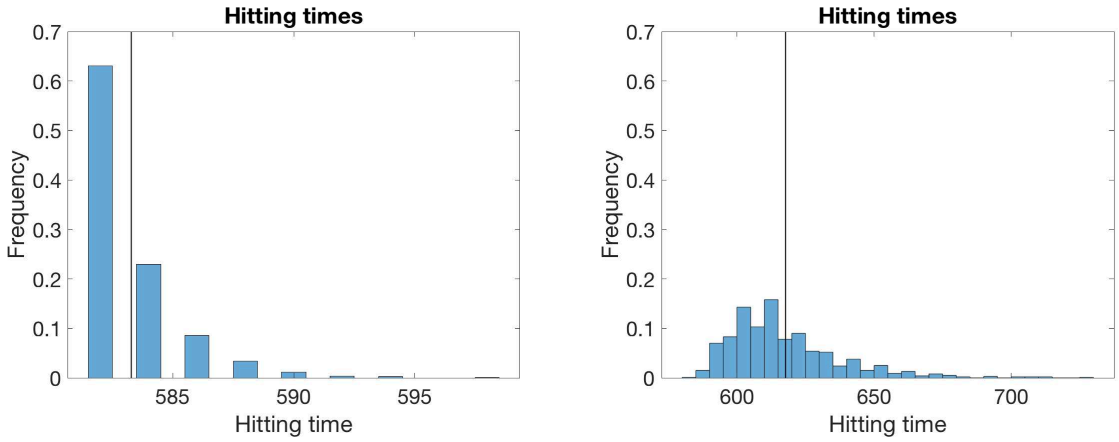


Figure 4. *Histograms of hitting times of an evolution of a random graph to a 2-community target graph. Each histogram corresponds to a rate parameter ($s = 1$, left; $s = 10$, right) and is based on 1000 trials. The corresponding analytic hitting times from Proposition 3.2 are shown in black lines; the empirical mean hitting times differ from them by less than 0.1%. The target distance d_t is set to 10.*

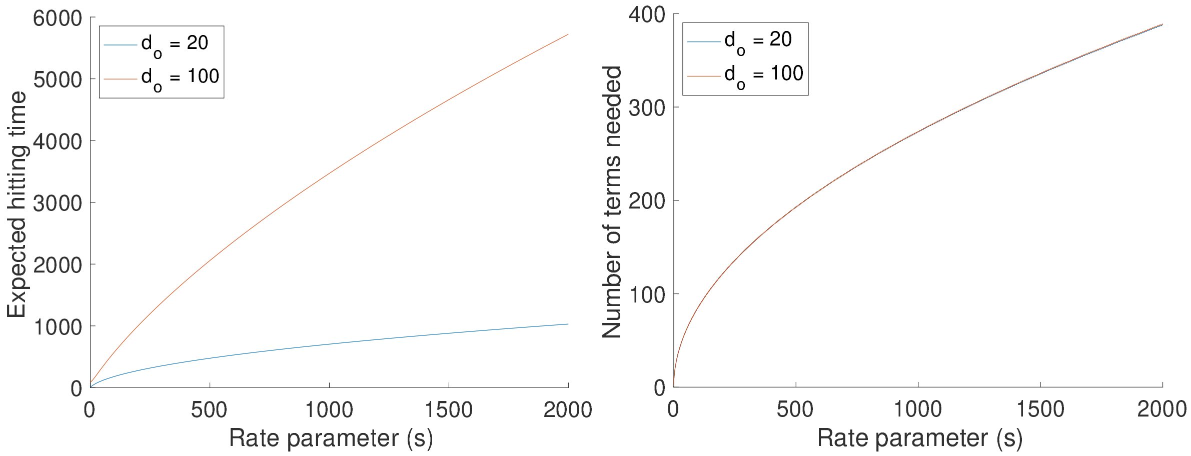


Figure 5. *Expected hitting time (left) and number of terms needed in the sum to reach machine precision (right) according to the analytic expression in Proposition 3.2, plotted as a function of the rate parameter s. The target edit distance d_t is set to 10, and two different initial edit distances d_o to the target graph are considered ($d_o = 20$ and $d_o = 100$).*

shows how the expected hitting time grows as a function of s and how the number of terms needed in (3.12) to reach machine precision grows as a function of s. If s is relatively small, then the terms in (3.12) decay rapidly, and the hitting time can be accurately approximated with a summation of a few terms regardless of the target or initial edit distance.

3.2.3. The Markov chain on the space of graphs.

The previous two results concerned a Markov chain on the space of edit distances, but our model is actually a Markov chain on the space of all possible graphs with a fixed number of vertices. Here we analyze the set of graphs that the model traverses. The exact transition probabilities between the graphs depends on how the advancing and regressing moves are chosen at each time step, but we can make some general statements. An n-vertex graph G can be viewed as a $\binom{n}{2}$-dimensional boolean vector indexed by pairs of nodes, where each entry of the vector indicates whether G contains the corresponding edge of the complete graph. Thus, the set of graphs can be viewed as the vertices of the $\binom{n}{2}$-dimensional hypercube, and the Markov chain on graphs can be viewed as a (biased) random walk on this hypercube. The edit distance between two graphs is the Hamming distance between their boolean vector representations, and we say that two graphs are adjacent if the edit distance between them is 1. Let the ith *layer* of the hypercube be the set of all graphs that are edit distance i from the target graph: The target graph forms layer 0, the graphs formed by deleting an edge or adding an edge to the target graph form layer 1, and so on. The farthest layer from the target graph is layer $\binom{n}{2}$, which consists of a single graph whose edges are precisely the ones that do not exist in the target graph.

Proposition 3.3. *Suppose that each advancing (resp., regressing) move is chosen uniformly at random from the set of all possible advancing (resp., regressing) moves. Then the time-averaged limiting distribution is uniform on all graphs in the same layer.*

Suppose that each advancing (resp., regressing) move is chosen uniformly at random from the set of all possible advancing (resp., regressing) moves, except that no false edges are allowed. Then the time-averaged limiting distribution is supported on the subgraphs of the target graph and is uniform on all such graphs that are in the same layer.

Proof. Note that a graph in layer i is adjacent to i graphs in layer $i-1$ and $\binom{n}{2}-i$ graphs in layer $i+1$. Thus, if p_i is the probability that a graph in layer i advances to a graph in the next lowest layer, then the Markov chain described in the first part of the statement can be described as follows: For $0 < i < \binom{n}{2}$, each graph in layer i has probability p_i/i of transitioning to an adjacent graph in layer $i-1$ and probability $(1-p_i)/\left(\binom{n}{2}-i\right)$ of transitioning to an adjacent graph in layer $i+1$. (The graph in layer 0 has probability $1/\binom{n}{2}$ of transitioning to each graph in layer 1, and the graph in layer $\binom{n}{2}$ has probability $1/\binom{n}{2}$ of transitioning to each graph in layer $\binom{n}{2}-1$.) The conclusion of the first part of the proposition holds because there is an automorphism of the Markov chain that sends a given vertex in a layer to any vertex in the same layer. (Any permutation of the vertices in layer 1 defines a permutation of the indices of each boolean vector and thus defines an automorphism of the hypercube graph.) For the second part, the graphs that contain edges not in the target graph are transient states in the Markov chain, and so, in the long term, the Markov chain looks like the chain described in the first part but supported on the subgraphs of the target graph. ■

4. Synthetic experiments.

We now investigate examples of our model where the target graph is structurally different from the starting graph. This experiment models forensic graph analysis—given structurally distinct starting and target graphs, can we build a sequence of graphs between them that captures the change and subsequently analyze the behavior of that sequence to detect the structural change? We use synthetic graphs in this section, as they have especially clear structure, but we consider real-data graphs in section 5.

We consider the stochastic block model [20], a widely used model for planted graph clustering. In this model, n vertices are divided into clusters and two vertices are independently connected by an edge with probability p if they are in the same cluster and smaller probability

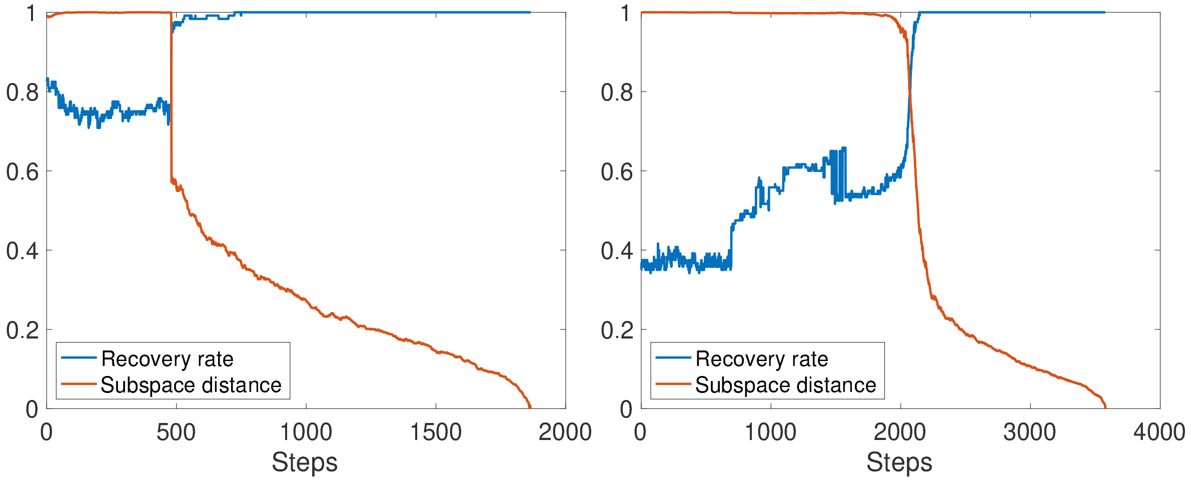


Figure 6. *Cluster recovery rate of a 2-block graph evolving into a 3-block graph in two different scenarios. The recovery rate under spectral clustering is shown in blue, while the subspace distance between the current eigenspace and the final eigenspace is shown in red. Left: One block splits into 2 blocks. Right: The 3 blocks in the target graph are independent of the 2 blocks in the starting graph.*

q if they are in different clusters. We analyze two scenarios where a starting 2-block graph evolves into a target 3-block graph. This allows us to explore how community structure affects the graph transition. In the first scenario, one block in the starting 2-block graph splits into 2 equally sized blocks of half the size. In the second scenario, the 3 blocks in the target graph are independently chosen and have no a priori relation to the 2 blocks in the starting graph.

For the experiment, we took $n = 120$ and set $p = 0.9$ and $q = 0.1$ for all the block structures. We ran both evolutions with $s = 1$ and target distance 0, and stopped the model as soon as it reached the target graph. (The target distance was 0, and so we stopped when the edit distance was first 0.) We measured the recovery rate (the fraction of nodes correctly classified) over time for the 3 clusters in the target graph using a spectral clustering algorithm [10]. We also measured the *subspace distance* between the space spanned by eigenvectors associated with the three largest eigenvalues of the symmetrically normalized adjacency matrix at each step and the analogous subspace of the symmetrically normalized adjacency matrix of the final target graph. The subspace distance between two subspaces represented by matrices U and V with orthonormal columns is $\sqrt{1 - \sigma_{\min}(U^T V)^2}$ [16]. This measures how well the dominant invariant subspaces of the current and target graph align and therefore if we expect a spectral algorithm to be able to pick up on the structure of the target graph.

Figure 6 shows the results of this experiment, and both scenarios display striking behavior. There is a point at which the subspace distance sharply declines, causing the recovery rate to sharply increase. The first scenario requires noticeably less time to detect the graph transition, presumably because the community structure of the target graph is related to the starting graph. Although the experiment is based on synthetic data, it suggests the capabilities of our model to transition from one structure to another and the availability of algorithms to detect structural changes as they occur.

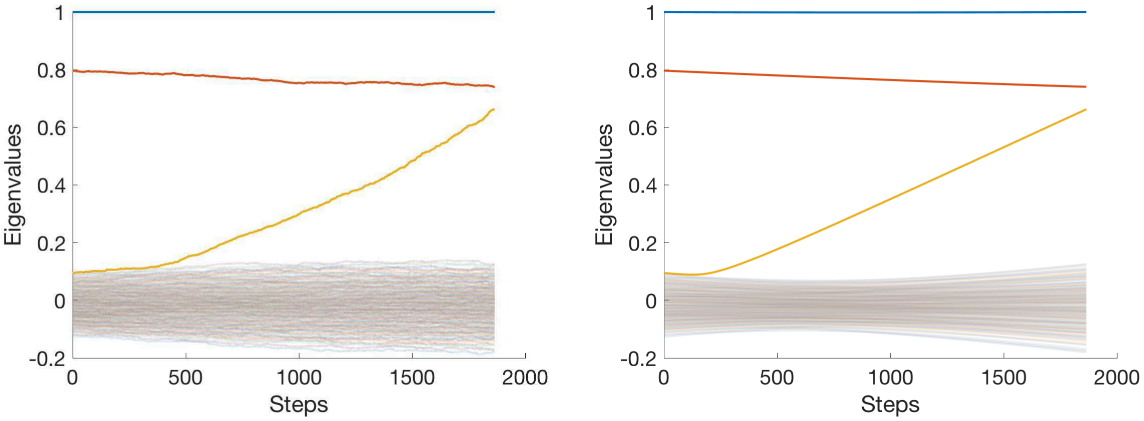


Figure 7. *Spectrum of the symmetrically normalized adjacency matrix of a 2-block graph evolving into a 3-block graph, where one block splits into 2 blocks. Left: Graph interpolation. Right: Equispaced linear interpolation between matrices. The full spectrum is computed at every step of the graph interpolation (resp., every increment of the equispaced linear interpolation), and the most significant eigenvalues are highlighted to distinguish them from the bulk.*

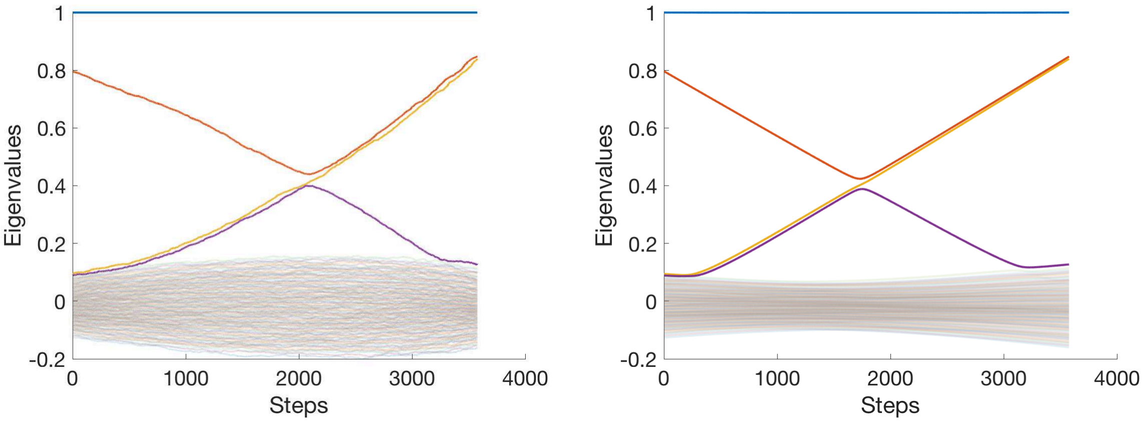


Figure 8. *Spectrum of the symmetrically normalized adjacency matrix of a 2-block graph evolving into a 3-block graph, where the 3 blocks in the target graph are independent of the 2 blocks in the starting graph. Left: Graph interpolation. Right: Equispaced linear interpolation between matrices. The full spectrum is computed at every step of the graph interpolation (resp., every increment of the equispaced linear interpolation), and the most significant eigenvalues are highlighted to distinguish them from the bulk.*

From the same experiment, we looked at the spectrum of the symmetric normalized adjacency matrix of the interpolating graphs in both scenarios and compared it to the spectrum of an algebraic interpolation between the normalized adjacency matrices of the starting and target graphs, where we ignore the graphs and simply interpolate linearly at equally spaced intervals between the two matrices. Figure 7 shows the resulting spectra for the first scenario, and Figure 8 shows the resulting spectra for the second scenario. In both figures, the spectra from the graph interpolation and linear matrix interpolation are strikingly similar, suggest-

Table 1
Summary statistics of undirected snapshot graph datasets.

Dataset	# snapshots	# nodes	# edges (max)
CollegeMsg [39]	2	1899	5852
email-Eu-core-temporal [40]	10	1005	16064
van de Bunt students [9]	7	32	59
DBLP coauthorship [32]	9	554885	407575

ing that our model can produce plausible spectral information while having the advantage of providing a sequence of graphs producing that spectral information. This is valuable since eigenvalue interpolation is in general known to be a difficult problem, especially in situations where the eigenvalues cross, as in the second scenario.

5. Real-data experiments.

We now interpolate between snapshots of several real-world networks, demonstrating that our model can provide a sensible graph interpolation where graphs and their properties vary "smoothly," even in the absence of any data between the snapshots. To interpolate between two graphs, we set the first graph to be the starting graph and the second graph to be the target graph. We then set the target edit distance to be 0 and ran our interpolation with a specified rate parameter until the target graph is reached. To interpolate between multiple graphs, we do this procedure for the first and second graphs, then the second and third graphs, and so on. We then compare our interpolation with extrapolation methods (growth models) as described below. Table 1 summarizes the datasets we use.

In our first type of example, we consider a strictly growing network and compare our interpolation with various extrapolations available from network growth models. For this purpose, we used the first two datasets shown in Table 1. The first one is based on private messages on a college messaging network [39]. The data consists of time-stamped edges indicating communication between two individuals. From this, we create a growing dynamic graph based on the accumulation of these messages over a period of time. We took two snapshots: The first snapshot is the empty graph, and the second snapshot is the aggregated graph of all communications within the first 30 days of the dataset.

Figure 9 shows the change in the mean and global clustering coefficient in the actual data over time compared with our graph model interpolations and three extrapolation methods. The mean clustering coefficient of a graph is the mean local clustering coefficient across all nodes, where the local clustering coefficient of a node is the number of connected pairs of neighbors of the node divided by the total number of pairs of neighbors (and is set to 0 if the node has degree 0 or 1). The global clustering coefficient of a graph is the number of closed length-2 paths divided by the total number of length-2 paths in the entire graph.

The interpolations in Figure 9 use rates $s = 1$ and $s = 1900$; the latter was chosen to match most closely with the real data. To elaborate further, we actually know that the real dynamic network took 21812 steps between the snapshots, so using Proposition 3.2, we took s to be the multiple of 50 such that the expected hitting time was closest to the number of steps taken. In general, a higher rate parameter increases the expected number of steps in the interpolation, so this parameter can be tuned if an estimate of the number of edits actually taken from one graph to another is known.

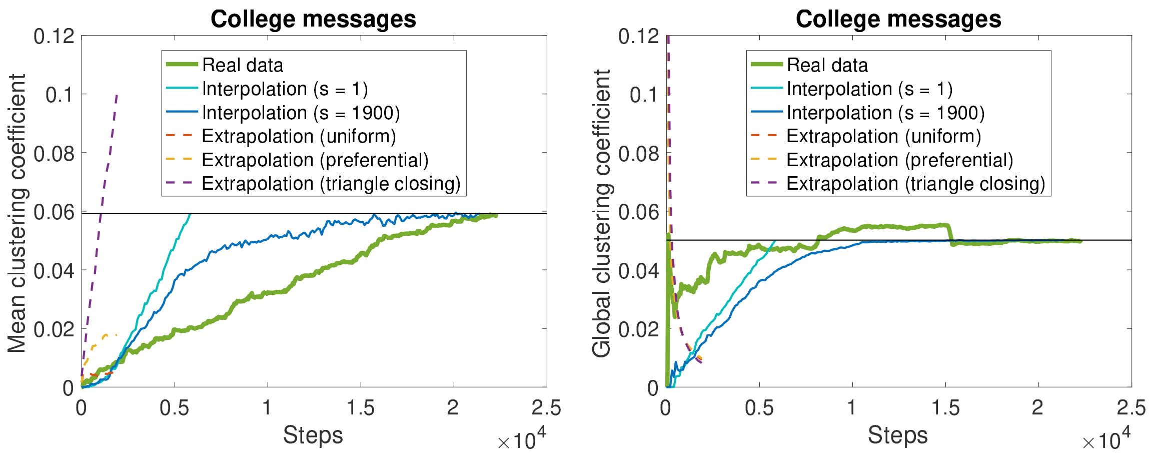


Figure 9. *Evolution of mean and global clustering coefficient in a college message network. The actual evolution is highlighted in green along with the interpolation and three extrapolation methods. The horizontal line indicates the clustering coefficient of the final snapshot. The interpolation with rate parameter $s = 1900$ corresponds most closely with the real data.*

The extrapolations are from three network growth models: uniform attachment, Barabási–Albert preferential attachment [6], and Jackson–Rogers triangle closing [22]. These growth models all start with a small initial graph. In each step, a new node appears and connects to a predefined number of nodes (on average) in the existing network according to a probabilistic distribution on the nodes. For uniform attachment, we start with a clique on some m vertices, and each new node connects to m nodes chosen uniformly at random in the existing graph. For preferential attachment, we start with a clique on some m vertices, and each new node connects to m samples of random nodes in the existing graph, chosen with probability proportional to their degree. For Jackson–Rogers triangle closing, we start with a clique on some $m_r + m_n + 1$ vertices. At each step, we pick m_r nodes in the existing graph uniformly at random and call them "parent nodes." The new node connects to each parent node independently with probability p_r. We also pick m_n nodes uniformly at random from the parents' immediate neighbors and connect to each of them independently with probability p_n.

We set the extrapolation parameters to match approximately the average degree of the target graph. For uniform and preferential attachment, the parameter m is the number of nodes that each new node connects to, and so we set m to be the nearest integer to the average degree of the target graph. For the college message network in Figure 9, $m = 3$, as can be seen from Table 1. The triangle-closing model has parameters m_r, p_r, m_n, and p_n. The average degree, in expectation, of a graph produced by this model is $m_r p_r + m_n p_n$, so we matched this with the average degree in the dataset. To set a reasonable choice of parameters, we set $p_r = p_n = 0.5$ and experimentally adjusted m_n relative to m_r until the mean clustering coefficient of the graph produced by the model approximately matched that of the target graph. Here we set $m_r = 5$ and $m_n = 1$ (we found that $m_r = 6$ and $m_n = 0$ produced a final mean clustering coefficient that was as low as the uniform attachment model).

Figure 9 shows that the extrapolation methods do not yield the correct final mean clustering coefficient or even the correct qualitative behavior for the global clustering coefficient, instead decreasing rapidly from 1 (corresponding to the starting clique with isolated vertices).

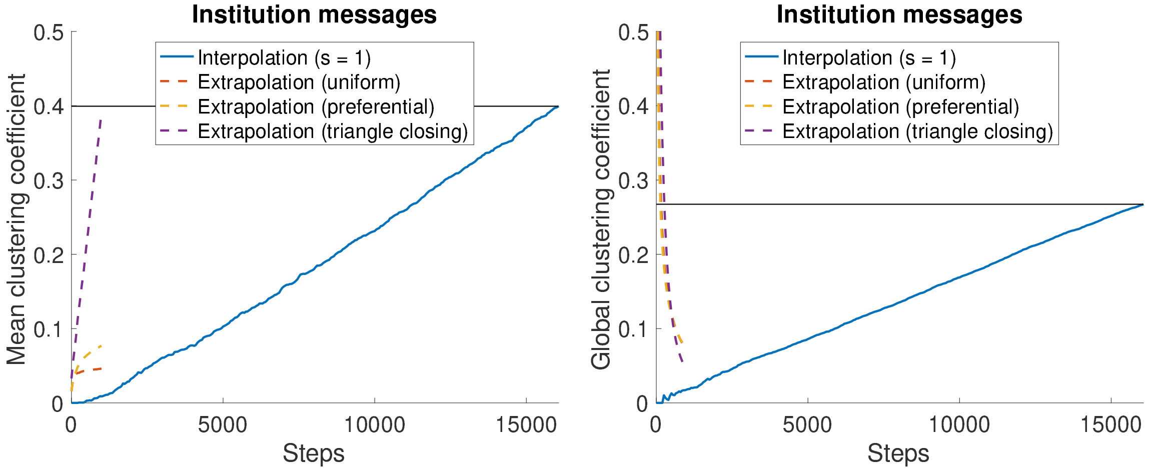


Figure 10. *Evolution of mean and global clustering coefficient on the email-Eu-core network using interpolation and three extrapolation methods. The horizontal line is the clustering coefficient of the final snapshot. Our interpolation better and more naturally fits the data.*

These existing methods are extrapolating from a starting graph, which is not appropriate for our task. This further motivates the use of our interpolation model.

We repeated our experiments using a dataset of e-mails at a European research institution [40] with time-stamped edges indicating e-mails exchanged between individuals at the institution. As before, we created a growing dynamic graph based on the accumulation of the e-mail exchanges. We did two experiments: one using just 2 snapshots (the empty graph and the final graph accumulating all the e-mail exchanges) and one using 10 snapshots (details below). For the first experiment, Figure 10 shows the change in mean and global clustering coefficient for our graph model interpolation with rate $s = 1$ and for the three extrapolation methods just described. For uniform and preferential attachment, we set $m = 16$, and for triangle-closing, we set $p_r = p_n = 0.5$, $m_r = 6$, and $m_n = 26$. Here we were better able to fit the triangle-closing model to match the final mean clustering coefficient of the target. However, for the global clustering coefficient, all of the extrapolation methods miss the mark.

Figure 11 shows our experiments on the same dataset but where we took a snapshot of the growing graph roughly every 100 days to get 10 snapshots in total. (Because the interval between time stamps is far from uniform, a number of snapshots are clustered near the end of the dataset when the time stamps are ordered chronologically.) For each interpolation, we used Proposition 3.2 to choose the best rate parameter up to a multiple of 50 since we know the edit distance between snapshots and the number of steps taken between them. For the extrapolations, we used the same models as before but added edges one at a time. As shown in Figure 11, only our interpolation produces a plausible result with roughly the correct overall number of steps, and it also best captures the statistical trends of the data over time. One apparent inaccuracy is the dipping behavior near the beginning of each interpolation. This may be due to the assumption of uniformly random edge edits, which degrade the clustering of the graph, as edges that get added near the beginning of the interpolation tend not to have any significant connection to the existing edges. This behavior is magnified in a later experiment (see Figure 14).

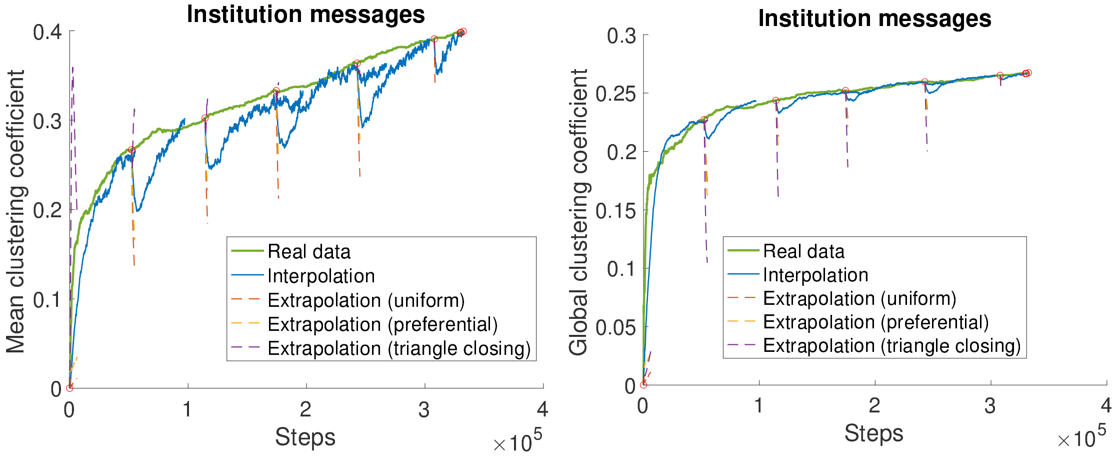


Figure 11. *Evolution of mean and global clustering coefficient on the email-Eu-core-temporal network. The actual evolution is highlighted in green along with the interpolation and three extrapolation methods. Red circles denote clustering coefficients of the snapshots themselves. The horizontal axis is according to the steps in the real data; this explains the overlapping lines in the interpolation since the number of steps in the interpolation does not align perfectly with the real data.*

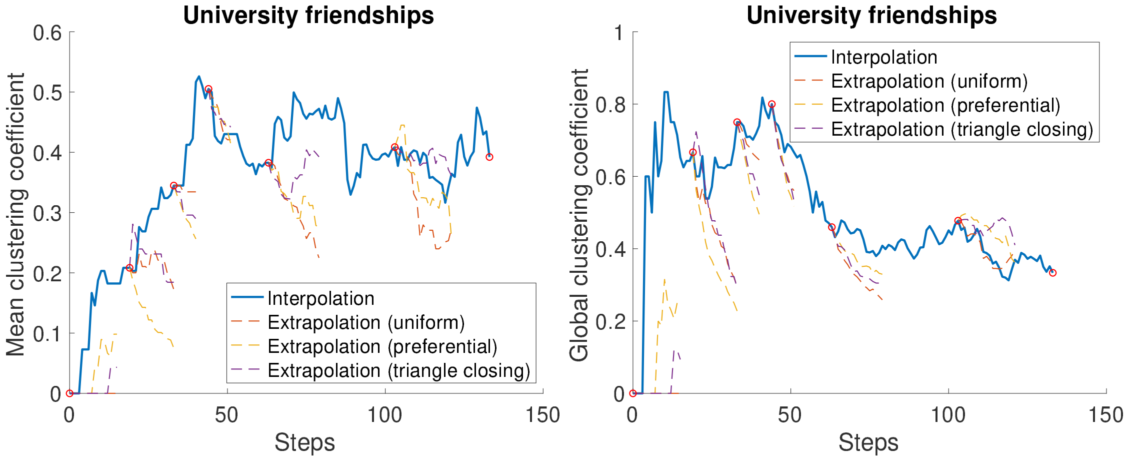
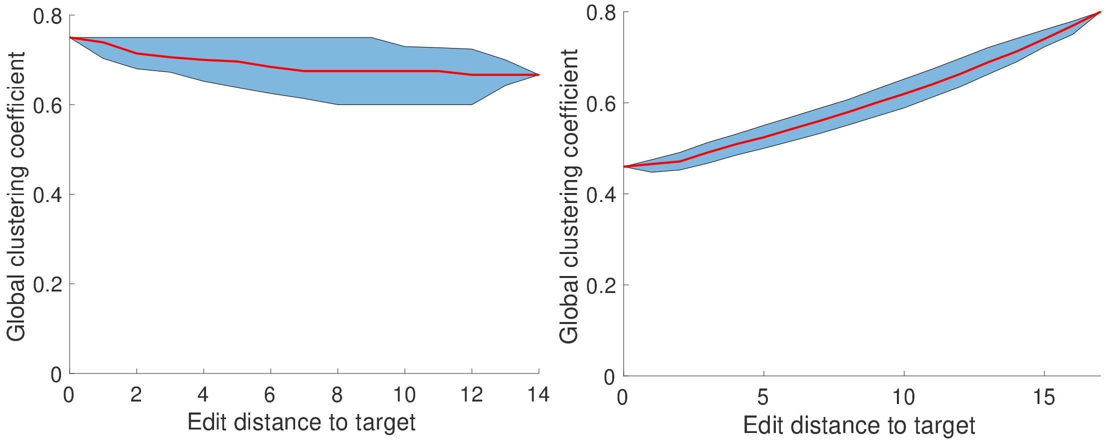


Figure 12. *Evolution of mean and global clustering coefficient in interpolation and three extrapolations between 7 snapshots of university freshmen friendships. The rate parameter $s = 1$ for the interpolation. Red circles denote clustering coefficients of the snapshots themselves.*

Our second type of real-world data example further shows the potential of our model to produce a feasible interpolation between a series of snapshots. The datasets we used here are the last two in Table 1. The first dataset consists of 7 snapshots of self-reported friendships between 32 university freshmen in a survey-based experiment [9]. Figure 12 displays the mean and global clustering coefficient over time for our interpolation with rate 1 and compares it with the same three extrapolation models. Although it is less natural to use extrapolation for nongrowing networks, we included a “decaying” component as follows. If the number of edges of the graph increased between snapshots, then we proceed as before with the usual extrapolation one edge at a time until the number of edges in the extrapolation equals the number of



(a) Snapshot 2 to 3.

(b) Snapshot 4 to 5.

Figure 13. *Quartile box plots of global clustering coefficient vs. edit distance of the interpolations between snapshots of university freshmen friendships with rate parameter* $s = 1$*. The quartiles are based on* 10000 *trials. The mean is in red and the lower and upper limits of the shaded region are the first and third quartiles, respectively.*

edges in the target snapshot. If, on the other hand, the number of edges of the graph decreased between snapshots, then in each step of the extrapolation, we select a "new node" uniformly at random. From this new node, we choose a neighbor to sever connections with. For uniform attachment, we delete a uniformly random neighbor; for preferential attachment, we delete a random neighbor inversely weighted by degree; and for triangle closing, we delete a random neighbor inversely weighted by the number of triangles in which the neighbor participates.

The interpolation and extrapolations in Figure 12 are shown for only one run, but the qualitative behavior across runs is similar. Notably, we observe that our interpolation scheme viably interpolates the snapshots, while the growth models fail to do so. For the purposes of verifying consistency across different instances of our interpolation scheme, Figure 13 provides a quartile box plot of the global clustering coefficient as a function of edit distance for two pairs of sequential snapshots: one where the beginning and ending clustering coefficients are similar and one where they are not. The mean clustering coefficient has a comparable level of variability.

We repeated the comparison of interpolation and extrapolation using the DBLP coauthorship dataset [32], which is a series of 18 graphs detailing the coauthor relationships among the conference proceedings papers on DBLP each year from 2000 to 2017. To prevent the clustering coefficients from being unduly influenced by outliers, we ignored papers from the dataset with more than 10 authors, and we aggregated the coauthorship data every 2 years (thus producing 9 snapshots). The comparison is shown in Figure 14.

In this case, the clustering coefficients for the interpolation dip substantially instead of staying roughly linear from one snapshot to another. This is because the edits in the interpolation are done uniformly at random. This presumably does not reflect the dynamics of the actual coauthorship dynamic network, which preserves the clustering structure where coauthorship forms cliques in the network. This coauthorship example shows that uniform edits are not a perfect model for describing changes in a network, and more correlated edit

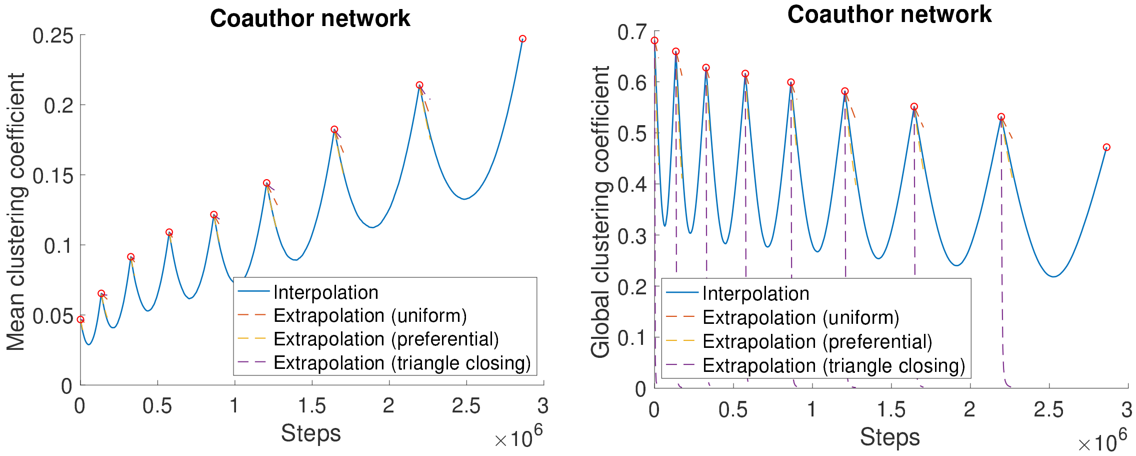


Figure 14. *Evolution of mean and global clustering coefficient in interpolation between* 9 *snapshots of coauthor relationships. The rate parameter* s *is* 1.

rules may be necessary to preserve such a high clustering coefficient. We leave this for future work.

6. Conclusions.

We developed a model for network interpolation, analyzed quantities of interest, and motivated it using snapshots from experiments. In doing so, we demonstrated substantial improvements over extrapolation methods in reconstructing important network statistics of dynamic graphs. We also provided a conceptual understanding of trajectories between graphs in terms of edits and edit distances and opened an area of investigation into the rules that should govern such trajectories. Although we analyzed the effect of the rate parameter in our model, further work remains to be done on the effect of sigmoid functions other than the standard logistic function and more complex edge edit rules beyond uniformly random edits. For example, edge edit rules that favor well-connectedness may help better preserve certain graph quantities of interest. Such insights would enhance the adaptability of our model to real-world situations.

We emphasize that our model has broad scope: It can be applied to any set of snapshots—real or synthetic—to generate random, plausible sequences of graphs to move from a starting graph to a target graph in a wide variety of domains. This makes our model applicable to generating synthetic streaming datasets with planted structure (including planted partition, planted clique, and planted coloring) as well as other types of emerging structure in real-world networks. It also has the potential to be used for on-demand or "live" data mining platforms. Furthermore, our model is amenable to theoretical analysis and opens up new theoretical directions in temporal network analysis.

REFERENCES

[1] R. Albert and A.-L. Barabási, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.

[2] R. Aldecoa and I. Marín, *Exploring the limits of community detection strategies in complex networks*, Sci. Rep., 3 (2018).

[3] M. Araujo, S. Papadimitriou, S. Günnemann, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra, *Com2: Fast automatic discovery of temporal ("comet") communities*, in Advances in Knowledge Discovery and Data Mining, Springer, New York, 2014, pp. 271–283, https://doi.org/10.1007/978-3-319-06605-9_23.

[4] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, *Group formation in large social networks: Membership, growth, and evolution*, in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, 2006, pp. 44–54.

[5] J. K. Ball, *Drug Abuse Warning Network,* 2006: *National Estimates of Drug-Related Emergency Department Visits*, Technical report, Substance Abuse and Mental Health Services Administration, Rockville, MD, 2007.

[6] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–12.

[7] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, *Simplicial closure and higher-order link prediction*, Proc. Natl. Acad. Sci., 115 (2018), pp. E11221–E11230.

[8] M. Bhattacharjee, M. Banerjee, and G. Michailidis, *Change point estimation in a dynamic stochastic block model*, https://arxiv.org/abs/1812.03090v1, (2018).

[9] G. G. V. D. Bunt, M. A. V. Duijn, and T. A. Snijders, *Friendship networks through time: An actor-oriented dynamic statistical network model*, Comput. Math. Organ. Theory, 5 (1999), pp. 167–192.

[10] A. Damle, V. Minden, and L. Ying, *Simple, direct and efficient multi-way spectral clustering*, Inf. Inference, (2018), https://doi.org/10.1093/imaiai/iay008.

[11] D. M. Dunlavy, T. G. Kolda, and E. Acar, *Temporal link prediction using matrix and tensor factorizations*, ACM Trans. Knowl. Discov. Data, 5 (2011), pp. 1–27, https://doi.org/10.1145/1921632.1921636.

[12] M. Farajtabar, Y. Wang, M. G. Rodriguez, S. Li, H. Zha, and L. Song, *Coevolve: A joint point process model for information diffusion and network co-evolution*, in Advances in Neural Information Processing Systems, 2015, pp. 1954–1962.

[13] J. H. Fowler and N. A. Christakis, *Dynamic spread of happiness in a large social network: Longitudinal analysis over* 20 *years in the Framingham Heart Study*, BMJ, 337 (2008), pp. a2338–a2338.

[14] L. C. Freeman and S. C. Freeman, *A semi-visible college: Structural effects on a social networks group*, in Electronic Communication: Technology and Impacts, Vol. 52, AAAS, Washington, DC, 1980.

[15] A. Ghasemian, P. Zhang, A. Clauset, C. Moore, and L. Peel, *Detectability thresholds and optimal algorithms for community structure in dynamic networks*, Phys. Rev. X, 6 (2016), 031005.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[17] A. Gorovits, E. Gujral, E. E. Papalexakis, and P. Bogdanov, *LARC: Learning activity-regularized overlapping communities across time*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery, ACM Press, New York, 2018, https://doi.org/10.1145/3219819.3220118.

[18] J.-D. J. Han, N. Bertin, T. Hao, D. S. Goldberg, G. F. Berriz, L. V. Zhang, D. Dupuy, A. J. M. Walhout, M. E. Cusick, F. P. Roth, and M. Vidal, *Evidence for dynamically organized modularity in the yeast protein–protein interaction network*, Nature, 430 (2004), pp. 88–93.

[19] K. Henderson, T. Eliassi-Rad, C. Faloutsos, L. Akoglu, L. Li, K. Maruhashi, B. A. Prakash, and H. Tong, *Metric forensics: A multi-level approach for mining volatile graphs*, in Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, 2010, pp. 163–172.

[20] P. W. Holland, K. B. Laskey, and S. Leinhardt, *Stochastic blockmodels: First steps*, Social Netwks., 5 (1983), pp. 109–137.

[21] P. Holme and J. Saramäki, *Temporal networks*, Phys. Rep., 519 (2012), pp. 97–125.

[22] M. O. Jackson and B. W. Rogers, *Meeting strangers and friends of friends: How random are social networks?*, Amer. Econ. Rev., 97 (2007), pp. 890–915.

[23] E. M. Jin, M. Girvan, and M. E. J. Newman, *Structure of growing social networks*, Phys. Rev. E, 64 (2001), 046132.

[24] M. Kivela, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, *Multilayer networks*, J. Complex Netw., 2 (2014), pp. 203–271.

[25] K. Komurov and M. White, *Revealing static and dynamic modular architecture of the eukaryotic protein interaction network*, Mol. Syst. Biol., 3 (2007), https://doi.org/10.1038/msb4100149.

[26] D. Kondor, I. Csabai, J. Szüle, M. Pósfai, and G. Vattay, *Inferring the interplay between network structure and market effects in bitcoin*, New J. Phys., 16 (2014), 125003.

[27] G. Kossinets, J. Kleinberg, and D. Watts, *The structure of information pathways in a social communication network*, in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, 2008, pp. 435–443.

[28] P. L. Krapivsky, S. Redner, and F. Leyvraz, *Connectivity of growing random networks*, Phys. Rev. Lett., 85 (2000), pp. 4629–4632.

[29] J. Leskovec and E. Horvitz, *Planetary-scale views on a large instant-messaging network*, in Proceedings of the 17th International Conference on World Wide Web, ACM Press, New York, 2008, https://doi.org/10.1145/1367497.1367620.

[30] J. Leskovec, J. Kleinberg, and C. Faloutsos, *Graphs over time: Densification laws, shrinking diameters and possible explanations*, in Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, 2005, pp. 177–187.

[31] K. Lewis, J. Kaufman, M. Gonzalez, A. Wimmer, and N. Christakis, *Tastes, ties, and time: A new social network dataset using facebook.com*, Social Netw., 30 (2008), pp. 330–342, https://doi.org/10.1016/j.socnet.2008.07.002.

[32] M. Ley, *DBLP*, Proc. VLDB Endowment, 2 (2009), pp. 1493–1500, https://doi.org/10.14778/1687553.1687577.

[33] P. Liu, A. Benson, and M. Charikar, *A Sampling Framework for Counting Temporal Motifs*, preprint, https://arxiv.org/abs/1810.00980, 2018.

[34] L. Michell and A. Amos, *Girls, pecking order and smoking*, Social Sci. Med., 44 (1997), pp. 1861–1869.

[35] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, *Growth of the Flickr social network*, in Proceedings of the First Workshop on Online Social Networks, ACM Press, New York, 2008.

[36] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, *Community structure in time-dependent, multiscale, and multiplex networks*, Science, 328 (2010), pp. 876–878, https://doi.org/10.1126/science.1184819.

[37] S. Navlakha and C. Kingsford, *Network archaeology: Uncovering ancient networks from present-day interactions*, PLoS Comput. Biol., 7 (2011), e1001119, https://doi.org/10.1371/journal.pcbi.1001119.

[38] J. Overgoor, A. R. Benson, and J. Ugander, *Choosing to Grow a Graph: Modeling Network Formation as Discrete Choice*, preprint, https://arxiv.org/abs/1811.05008, 2018.

[39] P. Panzarasa, T. Opsahl, and K. M. Carley, *Patterns and dynamics of users' behavior and interaction: Network analysis of an online community*, J. Amer. Soc. Inf. Sci. Technol., 60 (2009), pp. 911–932.

[40] A. Paranjape, A. R. Benson, and J. Leskovec, *Motifs in temporal networks*, in Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, ACM, New York, 2017, pp. 601–610.

[41] I. Scholtes, *When is a network a network? Multi-order graphical model selection in pathways and temporal networks*, in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, 2017, pp. 1037–1046.

[42] J. L. Snell and J. G. Kemeny, *Finite Markov Chains*, Springer, New York, 1976.

[43] T. A. Snijders, *Stochastic actor-oriented models for network change*, J. Math. Sociol., 21 (1996), pp. 149–172.

[44] K. S. Xu and A. O. Hero III, *Dynamic stochastic blockmodels for time-evolving social networks*, IEEE J. Sel. Topics Signal Process., 8 (2014), pp. 552–562.

[45] J.-G. Young, L. Hébert-Dufresne, E. Laurence, C. Murphy, G. St-Onge, and P. Desrosiers, *Network Archaeology: Phase Transition in the Recoverability of Network History*, preprint, https://arxiv.org/abs/1803.09191, 2018.