# Dynamics of networks: generation, dimensionality reduction, and coarse-grained evolution of graphs

Proposed by Alexander Holiday

under the supervision of

Professor Yannis Kevrekidis

11/01/2013

# Contents

# 1 Introduction

From collaborations among movie stars [2] to gene interactions in C. elegans [8], network science has found a vast array of applications in the past two decades. This is largely a result of the generality of the network framework: myriad systems are readily adapted to the network description; interacting bodies (e.g. a city, person, or protein) form nodes, the connections between them (e.g. via highways, Facebook friendships, or biological suppresion) create edges. Thus, one may usefully apply the same abstraction to study such disparate topics as the spread of opinions in a society and the kinetics of chemical reactions.

However, too often this merely leads to a recasting of the original problem. While this, in itself, can be useful, it fails to exploit the true power in such a formulation. There are several obstacles that prevent the realization of the full utility of networks, a selection of which will be the focus of this proposal. Broadly speaking, the generality of the construct is in some sense its undoing: while numerous problems are addressable, they require many different types of analysis. Thus, a researcher in neural networks may use none of the same tools as a civil engineer designing tranporation networks. Certainly, the dynamics between proteins in a cell and drivers in rush-hour traffic may share almost no similarities; however, this should not prevent investigators in these fields from having similar analytical tools at their disposal.

One of these methods may be considered a foundation upon which all others rest: the ability to computationally construct networks with desired properties. As the intimate link between a network's underlying structure and its resulting dynamics becomes increasingly evident [2] [3], it becomes difficult to justify the use of simple models in lieu of the ability to construct more accurate versions. Unfortunately, there is currently a lack of methods available to researchers for the construction of networks with desired properties. The algorithms that do exist tend to address a single variable of the network's architecture, leaving others unspecified. Therefore, work remains in creating more general algorithms capable of generating networks with a wide range of structures.

Even if one succeeds in accurately modeling a network's structure, the resulting flood of data can be overwhelming and unwieldily to analyze. To address this problem, a class of techniques have been formulated to extract various properties specifically from network systems. These range from community detection algorithms that suggest groupings of similar nodes, to methods that determine the ease with which a signal is propagated in a network [15] [17] [16]. While these have been effective tools in understanding network structure, their applicability is confined to the analysis of single networks. An avenue that has received relatively little attention, and one especially relevant to the study of dynamic systems, is formulating techniques that operate *across* networks. Extending existing methods to operate on *sets* of graphs has the potential to automate the discovery of important parameters governing the evolution of dynamic network systems, revealing their coarse descriptions. This would be a tremendous step forward in network analytics.

Developing methods that either create specific networks on demand, or determine important system characteristics is, itself, a valuable aim; however, by combining advances in both

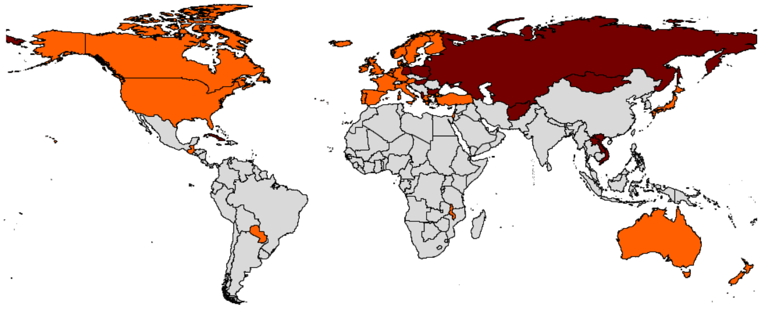


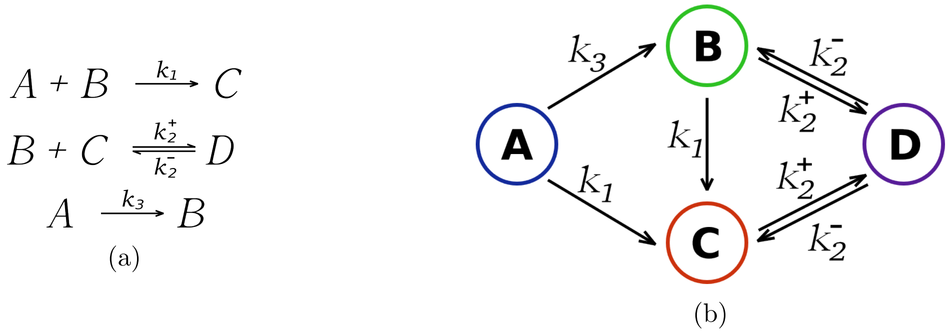
Figure 1: Grouping of countries based on United Nations voting records using a community detection algorithm [**?**].

abilities, we hope to enable various methods of coarse system analysis through *equation free* (EF) *modeling*. The EF framework contains a suite of analytical techniques, from coarse projective integration to systems-level optimization and design. In a field where networks may be composed of billions of nodes, the ability to examine behavior through a low-dimensional system is crucial in reducing simulation times, and allows a fuller analysis of dynamics such as determining fixed point stability and tracking bifurcations.

Graph generation, detection of signficant features across networks, and EF modeling of complex systems are all interesting avenues of research in their own right. It is hoped that the synthesis of these three elements will allow researchers in a broad range of fields to more accurately model real-world networks, and to attain useful information from subsequent results. We outline each of the three in sections 3, 4 and 5. Section **??** describes current advances in these topics, while section **??** concludes the proposal with a discussion of future research plans. Before continuing in this material, the notation used throughout the proposal will be specified.

## 2 Notation

A graph, $G$, is defined by a set of vertices (or nodes), $V(G)$, and the connections (or edges) between them, $E(G)$. The size of the network, $n$ $(= |V(G)|)$, is the total number of nodes, while the total number of edges is represented by $m$ $(= |E(G)|)$. A single vertex, $v_i$, is connected to another vertex, $v_j$ if and only if the corresponding edge, $e_{ij}$ is non-zero. An edge that begins and ends at the same vertex, $e_{ii}$, is called a loop. If $e_{ij} = e_{ji} \; \forall \; i, j$ then the graph is undirected, i.e. if $i$ is connected to $j$, $j$ must be connected to $i$, e.g. Facebook friendships. Otherwise it is directed. In many cases, the edges take binary values, $e_{ij} \in 0, 1$, and we call the graph unweighted. Alternatively we deal with weighted graphs in which the edge value may take any positive value, typically signifying the strength of connection between $v_i$ and $v_j$. A reaction network in which edges represent differential equations governing interactions between molecules would be a weighted, directed graph as the interactions between particles could be different, and particle A's influence on B would not imply a reciprocal influence by B on A. This is illustrated in 2.



(a)

(b)

Figure 2: Traditional chemical reaction equation (2a) and its equivalent reaction network (2b).

There are many ways of specifying a graph, the simplest being a list of all $v_i$ and $e_{ij}$ present. However, the most popular method of description is through an "adjacency matrix", a square, $n \times n$, matrix in which $A_{ij} = e_{ij}$. This form is especially helpful when the underlying graph is undirected ($e_{ij} = e_{ji}$) as the adjacency matrix is symmetric. An important property of vertices is their degree, given by $d_i = \sum\limits_{j=0}^{n} e_{ij}$. This measures a node's connectedness in the graph. From the definition, we see that in an unweighted, undirected graph, the degree of a vertex is simply the number of edges connected to it (with loops counted twice). Finally, as no strict definition of a network exists, we use the terms "graph" and "network" interchangeably. With this foundation, we continue with the original material.

# 3 Network Generation

Significant effort has been spent researching algorithms that create networks with different properties. By far the most popular method creates what is called an Erdős-Rényi random graph. Proposed in 1959 [10], it consists of the following simple procedure: given a set of vertices $V(G)$ and a probability of connection $p$, examine each possible edge in the graph (note there are $\binom{n}{2}$ possible edges in an undirected graph) and, with probability $p$, let $e_{ij} = e_{ji} = 1$, otherwise $e_{ij} = e_{ij} = 0$. Simply put, each edge exists with probability $p$. This straightforward method facilitates theoretical calculations, but, as one might guess, few real-world systems follow such a simplistic scheme.

Many different methods of graph generation have been proposed to more accurately capture network structure. They largely fall into two categories: those which describe an evolution of the graph to its final state, and those which generate graphs with specified properties. A famous example of the former is the scale-free preferential attachment model [2]. The method is as follows: begin with a small number of disconnected vertices, $n_0$. Add a new vertex, and connect it to $m \leq n_0$ of the pre-existing vertices. Let the probability that the new vertex, $v_{new}$, connects to one of the old vertices, $v_{old}$, be $P(e_{v_{new}v_{old}} = 1) = \frac{d_{old}}{\sum\limits_i d_i}$. Note that no graph properties have been specified *a priori*, except, trivially, the size $n$. In contrast, consider the Erdős-Rényi model just described. Here, the average degree, $\bar{d}$, is pre-specified, and a graph is constructed that adheres to this stipulation (note that the average degree and the probability of attachment, $p$, are related by $p = \frac{n\bar{d}}{2\binom{n}{2}}$). While the method of evolving a graph to its final form is useful when the underlying growth mechanism is understood, these models are specialized to the system under investigation. When information on the evolutionary process isn't available, as if often the case, it is more useful to measure certain features of the network in question and then use the property-specification class of methods to generate a graph with similar characteristics.

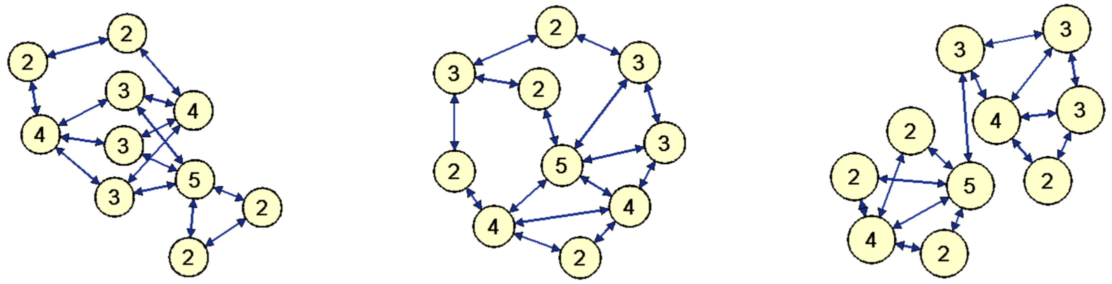
The number of such methods, and the variety of properties addressed by them, has grown steadily in recent years. From efficient procedures that generate Erdős-Rényi random graphs, to those that enable the simultaneous assignment of degree distribution and clustering coefficients [4] [7], a wide range of variables are tunable in the current collection of methods. The shortcoming of nearly all of the approaches in this toolset is that each addresses only a small number of specific properties. If the network under consideration has a certain degree distribution and contains a specific number of triangles, one must hope that an algorithm has been developed to deal exactly with the creation of a graph with a given degree distribution and triangle count. A method tailored to degree distributions cannot simply be combined with one that specifies the triangle count to achieve the desired outcome.

An innovative approach to this problem, the product of collaboration between the Floudas and Kevrekidis groups, aims to alleviate this drawback. Their technique, detailed in [13], is to formulate the problem as a task for mixed-integer linear optimization. Using this framework, they are able to add property specifications to the algorithm as desired. Each variable that needs to be addressed can be added as a building block in the overall formulation, making the approach tunable to the unique needs of each graph. Additionally, this method can guarantee the non-existence of certain graphs, a useful feature when detailing multiple complex properties. The downside of this wonderful generality is computational slowness. Searching the solution space of all possible graphs can become a daunting task at even modest network sizes of $n = 15$ if the searched-for graph is highly detailed. Using certain pre-processing steps, the speed has been greatly increased, but work remains if the method is to be applied to larger systems of $n > 100$.

Certainly, each algorithm has its unique strengths and weaknesses. While some can rapidly construct very large graphs of $n > 100,000$, they address only a few of the many potentially interesting properties of a network. The one method that allows users to add properties as desired suffers from scalability issues. The efficient creation of networks with different structural features remains an open area of investigation.

# 4 Data Mining across Networks

In the past decade, several factors have combined to fuel a wave of algorithms designed to analyze massive data sets. The internet has played no small role, as millions of users give information to online websites wittingly or otherwise, through user accounts or cookies. Additionally, the proliferation of electronic sensors in everything from cars to refridgerators has



(a) (b) (c)

Figure 3: Networks generated by the optimization framework, taken from [13]. The distribution of degrees is identical in both, but the clustering of the nodes was increased from 3a to 3c.

given companies access to a far greater variety and quantity of information. Crucially, too, spatiotemporal resolution of research experiments has become increasingly commonplace. In these scenarios, data-mining techniques are invaluable in extracting useful information from the resulting massive volumes of data. From the original workhorse of PCA, useful in finding data embedded in linear subspaces, to new, nonlinear manifold learning techniques such as DMAPS, today's techniques exhibit a wide range of approaches to dimensionality reduction [**?**]. However, these methods are not currently applicable *across* networks.

Generally speaking, the goal of a data mining algorithm is to take a collection of points in $\mathbb{R}^n$, $n \gg 1$ and embed them in $\mathbb{R}^m$, $m \ll n$, where the $m$ new dimensions capture the important details of the data. In many cases, data is easily amenable to such a description, e.g. a vector {*age, longitude, latitude*} for Amazon users. Unfortunately, when each data point is, itself, a network, there is no clear way to initially "pre-embed" each point into $\mathbb{R}^n$, and thus the host of techniques proven in other applications become useless. The two options are then to develop new techniques that specifically address the unique aspects of network data, or to devise a scheme that adapts established techniques to operate on graphs. We focus on the latter approach.

Many existing methods employ some measure of distance between points in their formulation. For example, a popular implementation of DMPAS requires a weight matrix $W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{\epsilon}}$, in which $x_i$ and $x_j$ are members of the dataset. Attempting to generalize vector or matrix norms might seem a natural place to start in defining distances between two graphs. As a norm, we would expect $\|x - y\| = 0 \Leftrightarrow x = y$; unfortunately, even determining whether two graphs are equivalent is a complete research puzzle unto itself, and is termed "the graph isomorphism problem".

## 4.1 Graph isomorphism

To understand the issue at hand, we must first discuss the distinction between labelled and unlabelled graphs. A labelled graph is one in which the vertices have been assigned a unique identification. In the case of a transportation network of highways between cities, the city names could function as labels. Then, if tasked with determining whether roadways changed between certain points in time, simply checking whether each city's connections remained the same would suffice. One the other hand, vertices in an unlabelled graph have no such intrinsic identity. Comparing two networks is no longer as easy as checking whether all connections to and from each node are the same, as there is no way to identify a node from the first graph, $v_{old}$, with its equivalent, $v_{new}$, in the second. To check whether two unlabelled graphs are different, all possible pairings of nodes from the two graphs may need to be considered. This is the **NP** problem of determining graph isomorphism, i.e. determining whether two graphs can be labeled in such a way as to make their connections equivalent. Formally, two graphs $G$ and $H$ are isomorphic if there exists a bijective function $f : V(G) \to V(H)$ such that $v_i \in V(G)$ and $v_j \in V(G)$ are adjacent if and only if $f(v_i) \in H(G)$ and $f(v_j) \in H(G)$ are also adjacent.

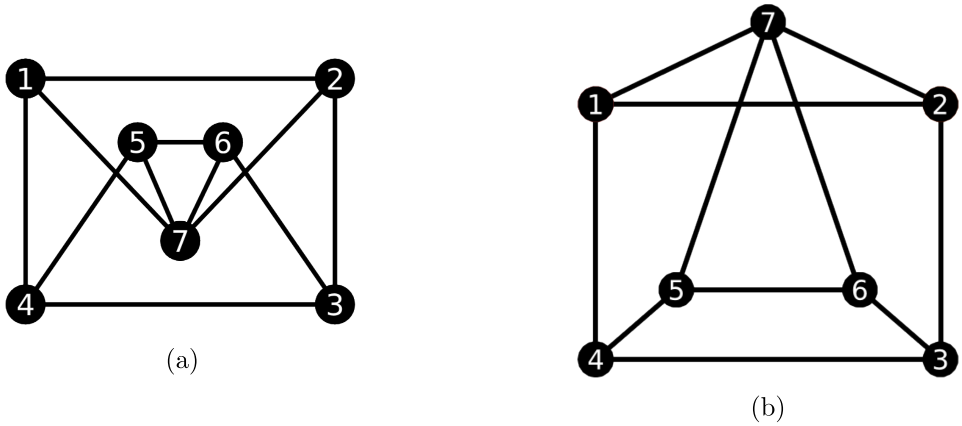


Figure 4: Different depictions of isomorphic graphs.

While a general polynomial-time solver is considered impossible by most as it would prove $\mathbf{P} = \mathbf{NP}$, approximation algorithms with guaranteed error bounds do exist. In fact, polynomial-time solutions are also available, but only for restricted classes of graphs.

## 4.2 Graph similarity

Given the difficulty in distinguishing whether two graphs are the same, the challenge in defining a computationally tractable distance measure is understandable. A novel approach to this issue would be to build off existing approximation algorithms. The hope would be to extend one of these methods to output some indication of the differences between graphs, and not just a boolean judgement of ismorphism. This modified method would then be used as the requisite distance measure in pre-existing data mining routines. If this innovative procedure fails, alternative methods have been established in the literature. These are detailed below.

### 4.2.1 Edit distance

The basis of this class of methods is the idea that a graph $G$ is similar to a graph $H$ if $G$ requires few changes, or "edit operations", to become identical to $H$. Edit operations consist of adding, removing and relabeling vertices, and adding and removing edges. Each action is assigned a cost, and the total cost of obtaining $H$ from $G$ defines their similarity. Unfortunately, determining the changes in $G$ required to obtain $H$ is computationally infeasible, so different approximations have been developed. In all existing methods, the graph is first converted into a string. This allows string edit distances to be used, which have a stronger theoretical foundation than edit distances of general graphs [11].

The process of seriation itself is varied, though many approaches use the eigenvector corresponding to the second smallest eigenvalue, i.e. the Fiedler vector, of some transformation of the adjacency matrix. A popular approach frames the problem as the minimization of

$$g(\pi) = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} A(i,j)(\pi(i) - \pi(j))^2$$

for some permutation $\pi$ [1]. As minimizing the above is **NP-hard**, the discreet formulation is relaxed in application. Once string representations are determined, any method of string comparison may be used, from string kernels to Dijkstra's algorithm based methods [23] [20].

### 4.2.2 Maximal common subgraph

Here, the distance between $G$ and $H$ is defined as

$$d(G,H) = 1 - \frac{|mcs(G,H)|}{max(|G|,|H|)}$$

where $mcs(G,H)$ is the maximal common subgraph between $G$ and $H$, that is, the largest subgraph of $G$ isomorphic to a subgraph of $H$. This straightforward formula doesn't require the subjective cost function of the edit distance, and is provably a metric on the space of graphs [6]. Its computation is, however, **NP-hard**. Nonetheless, heuristics have been employed to use the method in matching chemical structures [19].

### 4.2.3 Graph kernels

A kernel $k(x, x')$ must be symmetric and positive semi-definite; thus, kernels are a less restrictive class of operations compared to metrics, allowing greater freedom formulation. Relatively few graph kernels have been created. Those that have employ random walks over the vertices and edges in their formulation. They have recently been shown to take the general form:

$$k(G,H) = \sum_{k=0}^{\infty} \mu(k) q_{\times}^T W_{\times}^k p_{\times}$$

where $q$ and $p$ are the initial distribution of random walkers and the stopping probabilities, respectively, $k$ is the number of edges traversed during the walk, $W$ represents a sort of combination of $G$ and $H$ known as a product graph, and $\mu(k)$ is a weighting to ensure the sum converges. By defining $\mu(k) = \lambda^k$ for some $\lambda$, the infinite sum reduces to the problem of inverting $(\mathbf{I} - \lambda W_\times)$. This can be computed efficiently in a variety of ways, from conjugate gradients to spectral decompositions [5]. The downside of these methods is that the value of $\lambda$ defining $\mu(k) = \lambda^k$ is often taken to be very small. This ensures convergence of the sum, but neglects higher order terms, i.e. longer walks. The resulting comparison of short walks only captures the local network topology. Defining kernels based on other properties may permit a more holistic measure of graph similarity.

# 5 Equation Free Modeling

Dynamic network models prescribe behavior at the level of individual vertices and edges, enabling researchers to construct systems with an incredible level of detail. However, while we seek to understand the long-term macroscopic dynamics of the networks, explicit, accurate equations governing this system-level evolution are typically unavailable or poorly understood. To circumvent this need for macroscopic equations in system-level analysis, we turn to *equation free* (EF) modeling. However, there are two prerequisites to implementing the EF framework: an appropriate macroscopic description of the network, along with some method of translating these macroscopic variables into realizations of a microscopic system. Advances in the previous sections 3 and 4 would fulfill both.

## 5.1 Coarse time-stepper

The coarse time-stepper is a basic component of EF modeling, providing the evolution of coarse variables over a short time interval. This is accomplished through controlled initialization of short periods of fine-scale simulations. Define the high- and low-dimensional spaces as $F$ and $C$ (fine and coarse). To move between these two levels of system description, we define the restriction operator $\mathbf{R} : F \to C$, e.g. an extended DMAPS method, and the lifting operator $\mathbf{L} : C \to F$, e.g. some graph generation algorithm. It is important to note that the variables of $C$ may not always provide an accurate representation of the system's fine-scale behavior. Whenever a full simulation is initiated, high-dimensional variables may not be immediately slaved to the low. Instead, a period of "healing" is required, after which the system evolves on a slow manifold in $C$. We define this healing interval as $h$. The coarse time-stepper then proceeds as follows:

1. Initialize a fine-scale simulation at the desired point $u(t_0) \in F$.

2. Run the fine-scale simulation to $u(t_0 + h)$. Behavior can now be described with the low-dimensional variables in $C$.

3. Continue fine-scale simulation, using $\mathbf{R}$ to record $n$ collections of coarse variables at intervals of $\Delta t$.

This provides a time-series of macroscopic variables $\{\mathbf{R}(u(t + k\Delta t))\}_{k=0}^{n}$. With the ability to investigate low-dimensional dynamics at will, more powerful EF analysis becomes available.

### 5.2 Coarse projective integration

Coarse projective integration (CPI) exploits the assumed smoothness of the coarse variables' evolution to accelerate simulations. Define the coarse variable $U \in C$. The procedure is outlined as:

1. Run the coarse time-stepper to record the coarse timeseries $\{U_t\}_{t=t_0}^{t_f}$.
2. Using this collection, numerically calculate $\frac{dU}{dt}(t_f)$.
3. With some integration routine, e.g. Euler' method, project $U$ forward to some $U_{new} = U_{t_f+\Delta t}$.
4. Lift $U_{new}$ to $u = \mathbf{L}(U_{new})$.
5. Iterate these previous steps until the desired system state is reached.

By projecting forward the few variables that evolve smoothly on the system's slow manifold, expensive, full simulation is significantly reduced. The result is a steep improvement in computation time.

### 5.3 Coarse fixed point calculations

Given a macroscopic state $U_n \in C$, let $U_{n+1} = \Phi_T(U_n)$ be the state of the system after time $T$. A fixed point could be evaluated by locating $g^*$ such that

$$F(U^*) = U^* - \Phi_T(U^*) = 0$$

Typically, as in Newton-Raphson iteration, the derivatives $\frac{dF}{dU}$ would be needed to update the solver. While certainly not available analytically, these derivatives can be approximated numerically. This is an effective but expensive task, as it requires the repeated use of the coarse time-stepper. Using fixed point methods like the generalized minimization of residuals (GMRES) would improve performance [21].

### 5.4 Coarse bifurcation analysis

Once a steady state has been found, we may track its dependence on system parameters to analyze system bifurcations. This can be accomplished, for example, by a pseudo-arclength continuation scheme. Define $\lambda$ as the system parameter and $s$ the arc-length along the $\|U\|$ vs. $\lambda$ bifurcation curve, i.e. $s$ parameterizes $U(s)$ and $\lambda(s)$. Then, if we know the solution at some point $s_0$, we may solve the following system to find the values of $U(s_0 + ds)$ and $\lambda(s_0 + ds)$:

$$F(\mathbf{U}, \lambda) = 0$$

$$\|\frac{d\lambda}{ds}(s_0)\|^2 + \|\frac{d\mathbf{U}}{ds}(s_0)\|^2 = 1$$

# 6 Current Work

Initial research has focused on accelerating simulations of dynamical network systems. While leaping to the third of three steps may seem premature, it quickly reveals weaknesses in the overall process of simulation, and serves as a guide for future investigation. Two cases were studied, a voting model with possible applications in sociology, and an edge reconnecting model, which mainly serves as a toy mathematical network for which behavior can be derived analyticaly. After briefly describing the dynamics of each, current progress in simulation acceleration will be discussed.

## 6.1 Voting model

The $k$ opinion voter model initally consists of an Erdős-Rényi graph of size $n > 10,000$, with small average degree, $\bar{d} \asymp 1$. To begin, each vertex is randomly assigned an opinion based on some initial distribution $\{p_i\}_{i=1}^k$. With this initial state and a "probability of reattachment" $\alpha$, the graph evolves as follows (note that $\zeta(v_i)$ denotes the opinion of $v_i$):

1. Choose an edge $e_{ij}$ uniformly at random from $E(G)$.
2. Randomly choose one of the endpoints of $e_{ij}$, call it $v_i$. The other endpoint will be $v_j$.
3. Repeat steps one and two until $\zeta(v_i) = \zeta(v_j)$, i.e. the vertices' opinions do not match.
4. With probability $\alpha$, remove $e_{ij}$ from the graph.
   (a) Choose a new vertex $v_k$ uniformly at random from $V(G)$ until $e_{ik} \notin E(G)$.
   (b) Add $e_{ik}$ to $E(G)$.
5. Otherwise, with probability $1 - \alpha$, set $\zeta(v_i) = \zeta(v_j)$.

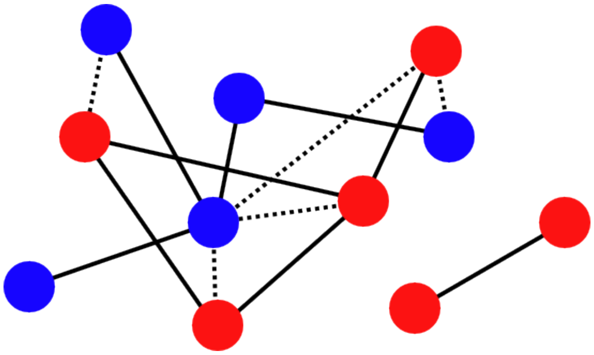


Figure 5: A possible configuration of a two-opinion model. The dotted lines represent conflicts. We may take the blue nodes to be Democrats, the red Republicans; then, the highly connected blue node may represent President Obama, the isolated red component Michele Bachmann and Sarah Palin.

This process is iterated until the system reaches a consensus state in which $\zeta(v_i) = \zeta(v_j)\ \forall\ v_i, v_j \in V(G)$, i.e. every edge connects vertices of the same opinion. An edge $e_{ij}$ is called a conflict whenever $\zeta(v_i) \neq \zeta(v_j)$. Thus the system is evolved until all conflicts are removed. While the general $k$ opinion model has received some attention [22], our focus will be the simpler two opinion model. In this case, it has been shown that, for a certain initial minority fraction $p_{minority} \leq 0.5$, a bifurcation occurs as $\alpha$ increases. At low values of $\alpha$, fewer edges are rewired and the system approaches a static state. In this limit, the system should converge once each of the $m$ edges are selected, which occurs in $O(n\sqrt{n})$ steps [9]. As $\alpha$ increases, so does The frequency of rewiring edges. This slows the rate of consensus to $O(n^2)$. Interestingly, besides differences in convergence speed, the final consensus state also changes. In the low-$\alpha$ limit, the final minority fraction is an increasing function of $\alpha$. In the slower, high-$\alpha$ range, the final minority fraction is unchanged from its initial value. Fig. **??** illustrates the effect of $\alpha$ and $p_{minority}$ on the final consensus state.

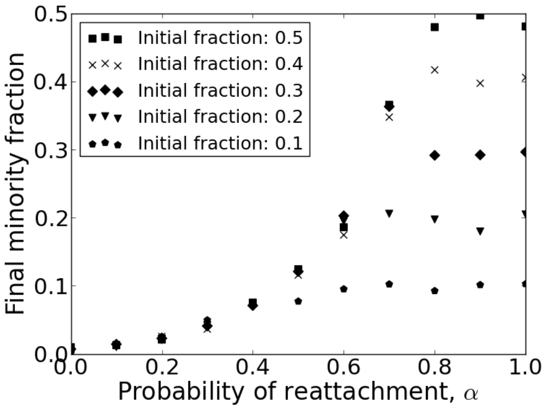
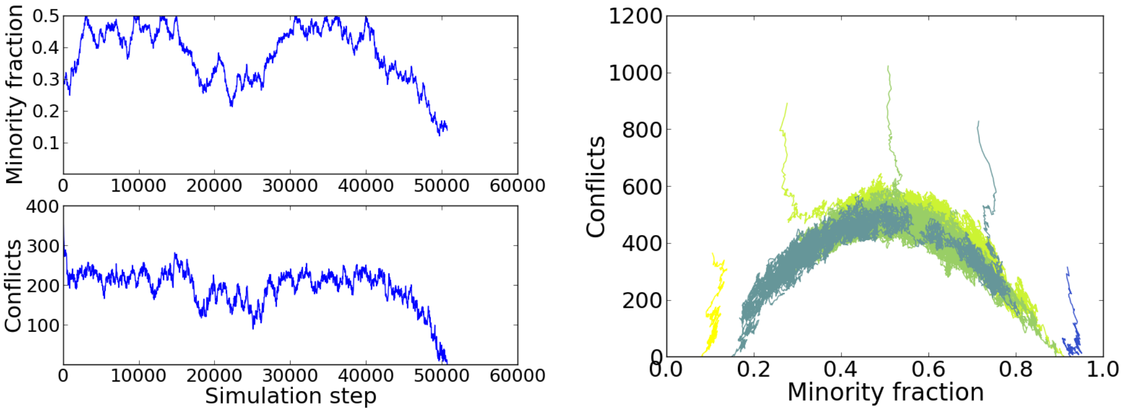


Figure 6: The effects of $\alpha$ and initial minority fraction on the final state. Above $\alpha \approx 0.8$, the final and initial fractions are approximately equal. Below this, a curve is mapped out. Any initial minority fraction above the curve will evolve towards the steady state value below.

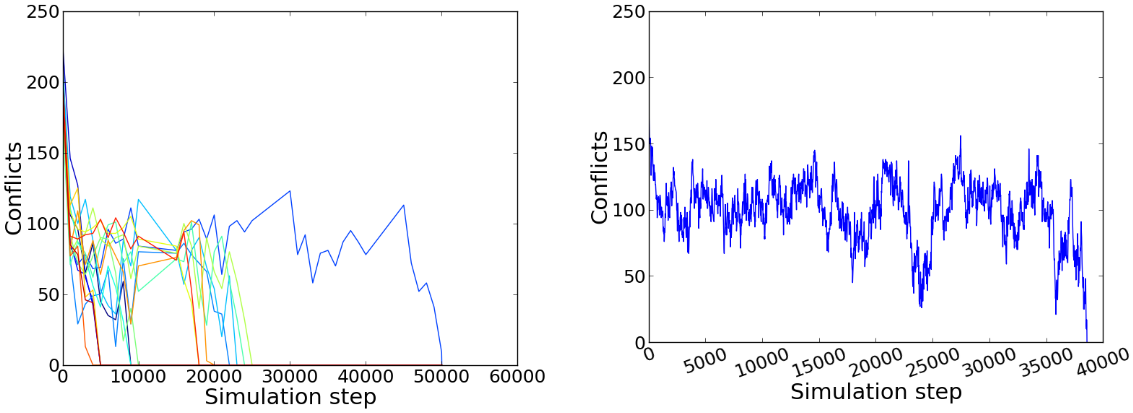
By creating phase plots of different system properties, it was observed that the minority fraction dictates the state of the system. Fig. **??** illustrates that the number of conflicts could be written as a function of the minority fraction, while **??** shows the independence of initial and final minority fractions. This suggested that the system minority fraction could serve as a macroscopic variable in CPI. The system is stochastic; thus, in order to obtain smooth evolutions of variables, the average trajectory of an ensemble of voting models was used. In fact, both the minority fraction and conflict count were used as coarse variables. After projecting these variables forward in macroscopic space, it remained to generate a graph with a specific minority fraction and conflict count. Thankfully this was a simple task, and a customized algorithm was developed. CPI results are shown in Fig. **??**.



(a)

(b)

Figure 7: Evolution of minority fraction and number of conflicts, 7a, and phase plot, 7b. Note the strong correlation between the minority fraction and conflict evolution in 7a.



(a) CPI in an ensemble of 16 simulations

(b) Direct simulation

Figure 8: System evolution with and without CPI. Note that initially in 8a, with many trajectories averaged together, a large number of models quickly converge. However, the remaining runs become somewhat erratic as they lose the dampening effect that averaging with many simulations had provided. In both cases, $n = 200$.

### 6.2 Edge reconnecting model

The edge reconnecting model, proposed in [18], presents special difficulties in creating accurate low dimensional representations of the overall system. Namely, the network is allowed to have multiple edges connecting the same vertices. Such a construction is termed a “multigraph”. Initially, $m \asymp n^2$ edges are distributed uniformly among the $n$ vertices. Then the multigraph evolves as a Markov chain according to the following dynamics:

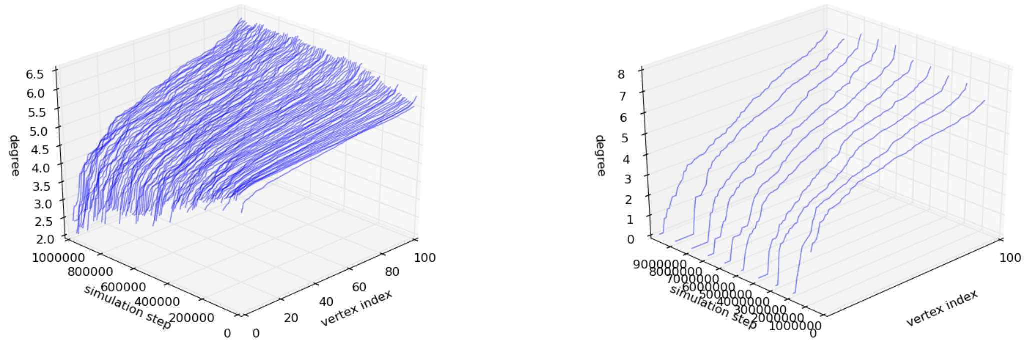
1. Choose an edge $e_{ij} \in E(G)$ uniformly at random and flip a coin to label one of the ends as $v_i$
2. Choose a vertex $v_k$ using linear preferential attachment: $P(v_k = v_l) = \frac{d_l}{\sum\limits_{i=1}^{n} d_i}$
3. Replace $e_{ij}$ with $e_{ik}$

This process is repeated until a frozen state is reached, at which point the degree distribution remains constant.

Interestingly, while the network is strictly unlabeled, the preferential attachment dynamics tend to stratify the vertices into a low degree majority and extremely high degree minority. This permits a sort of pseudo labeling of the vertices, as a vertex of high (low) degree at time $t_1$ is likely to retain its high (low) degree at $t_2$. Therefore, when sorted by degree, the evolution of the adjacency matrix appears smooth, as shown in Fig. 10b.

Two distinct timescales arise from these dynamics, $T \asymp n^2$ and $T \asymp n^3$ where $T$ is the number of steps. On the faster, $O(n^2)$ scale, the degrees of the vertices may be considered constant, while the number of parallel edges between vertices changes. On the slower $O(n^3)$ scale, the degree distribution evolves to a steady state value. While these separate timescales have been proven in [18], identifying them through numerical simulations is complicated by a couple of factors. First, the exact timescales themselves are difficult to discern. Both scales are really $O(\rho_1 n^2)$ and $O(\rho_2 n^3)$, where the constants $\rho_i$ are evaluated at the limit of infinite-sized graphs, $n \to \infty$. This hints at the second, larger, problem: many of the results on the existence of these timescales in the first place are only valid in this large-$n$ limit. Simulation time then becomes problematic. Figs. 9a and 9b illustrate attempts to visualize these separate scales of evolution. The changes in each appear quite gradual, and no distinct timescales are evident.

Our approach in coarse-graining system dynamics is based on the existence of a gap in the

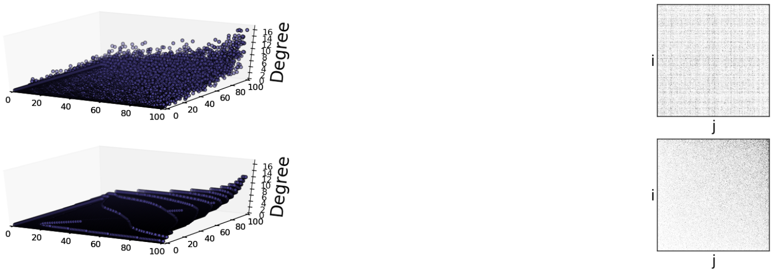


(a)

(b)

Figure 9: Degree evolution on $n^2$ and $n^3$ timescales, in 9a and 9b respectively.

spectrum of the adjacency matrix, and the subsequent ability to approximate $A \approx \lambda_1 v^{(1)} v^{(1)\,\dagger}$ where $A$ is the adjacency matrix of the system, $\lambda_1$ is the leading eigenvalue of $A$, and $v^{(1)}$ the corresponding eigenvector.



(a) Reconstruction of original adjacency matrix (top) using first eigenvalue/eigenvector pair (bottom).

(b) The unsorted adjacency matrix appears random (top), but sorting by vertex degree reveals its structure (bottom).

Figure 10: Reconstruction of adjacency matrix, 10a, and vertex sorting, 10b.

Fig. 10a illustrates the reconstruction of the adjacency matrix as $A_{ij} = \lambda_1 v_i^{(1)} v_j^{(1)\,\dagger}$ (after multiplication, each entry $A_{ij}$ was rounded to the nearest integer if $i \neq j$ or to the nearest even integer if $i = j$). Visually, the two correspond very well.

As $\lambda_1 v^{(1)} v^{(1)\,\dagger}$ appeared a good approximation of $A$, it was reasoned that we could use this eigenvalue/eigenvector combination as a coarse description of the system. In order to further reduce dimensionality, the eigenvector was fitted with a fifth-degree polynomial. Five degrees were used as additional terms didn't significantly increase fitting accuracy. The six coefficients of this function were then used as a smaller set of coarse variables, leading to a final seven-dimensional representation of the system (six coefficients plus an eigenvalue). The following outlines the CPI framework:

1. Simulate the full edge reconnecting model dynamics for some number of steps until the fast variables are sufficiently slaved to the slow.

2. Record the adjacency matrix as the system evolves on the slow manifold.

3. Project forward the coarse variables (coefficients and eigenvalue).

4. Reconstruct a new adjacency matrix from the new, projected coefficients and eigenvalue:

   (a) Compute a new eigenvector as $v(i) = \sum_{k=0}^{k=6} i^k c_k$ (where $c_k$ represents the coefficients of the polynomial and $v(i)$ the $i^{th}$ component of $v$) and round to the nearest integer

   (b) Compute the new adjacency matrix as $A_{ij} = \lambda_1 v_i^{(1)} v_j^{(1)\,\dagger}$ and round as discussed previously

5. Repeat from step one until system reaches steady state

Preliminary results of this method are shown in Fig. 11, in which the evolution of the degree distribution is shown for both the full simulation and a simulation in which CPI has been employed. The two generally coincide.

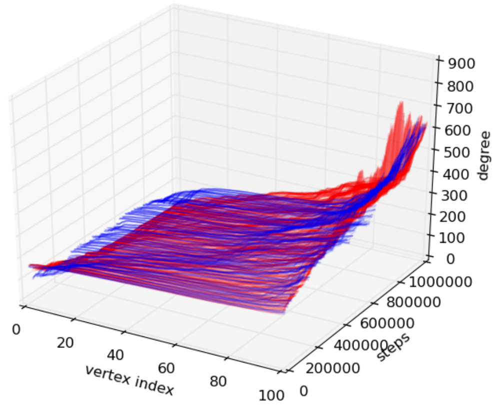


Figure 11: Overlay of direct simulation (red) and CPI (blue).

# 7 Future Work

A number of areas present themselves as good research directions. They're divided by subject below.

## 7.1 Graph generation

We will start by re-examining the recent optimization based method of graph generation. The main limitation of the approach is its scalability. Preprocessors have been shown to significantly reduce runtimes, but have been created to address specific properties [12]. These operate by removing search paths unlikely to yield the specified graph before the optimization algorithm begins. This works particularly well when many distinct (non-isomorphic) graphs exist that satisfy the stipulations. In this case, the preprocessor does not need to be particularly careful in its deletions, as it is unlikely to remove all branches leading to feasible solutions. When more specificity is needed, it is possible to tune this step to be more cautious in its selections. Creating a generic preprocessor for arbitrary properties would retain the generality of the method while increasing speed.

### 7.2 Data mining across networks

This nascent research topic has received little attention. Certain sub-problems, e.g. graph isomorphism, graph matching, and vertex matching, have been thoroughly investigated; but, the concept of applying dimensionality techniques when data points are, themselves, networks seems new. As mentioned, it may be possible to adapt an existing approximation algorithm that determines graph isomorphism to additionally yield information on the differences between two networks. A thorough understanding of the structure of these methods through an extensive literature review will be a necessary starting point.

Alternatively, the computer science community has established a number of graph similarity methods, as outlined in 4.2. If these test poorly, the most promising direction wouldappear to be in defining new kernels between graphs. The foundation of these graph kernels is a paper from 1999 [14] which provides a general framework from which other kernels could be invented.

In general, work would focus on developing graph similarity measures in simple models, such as Erdős-Rényi, in which issues would be more readily diagnosed and guarantees of success or failure more easily proven. Additionally, simpler, heuristic measures of graph similarity have been shown to work in certain scenarios. These could be used while more grounded methods are developed, or a combination of simple measures could be found that adequately addresses our needs.

### 7.3 Equation free modeling

The immediate aim in this direction will be to apply diffusion maps to the voting model data in an attempt to recover the minority fraction as a good coarse variable. This is expected to work, as DMAPS has been able to discover appropriate coarse variables in other collections of graphs using heuristic similarity measures. It would also be useful to apply the coarse fixed point calculations to the edge reconnecting model, as direct simulations are computationally intensive.

## References

[1] J. E. Atkins, E. G. Bomanl, and B. Hendrickson. A Spectral Algorithm for the Seriation Problem. In *Symposium on theory of computing*, number November, 1994.

[2] A. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(October):509–512, 1999.

[3] B. Barzel and A.-L. Barabási. Universality in network dynamics. *Nature Physics*, 9(9):1–9, Sept. 2013.

[4] V. Batagelj and U. Brandes. Efficient generation of large random networks. *Physical Review E*, 71(3):036113, Mar. 2005.

[5] K. M. Borgwardt, I. R. Kondor, and N. N. Schraudolph. Graph Kernels. *Journal of Machine Learning Research*, 9:1–41, 2008.

[6] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, Mar. 1998.

[7] M. Deijfen and W. Kets. Random intersection graphs with tunable degree distribution and clustering. *Probability in the Engineering and the Informational Sciences*, 2009.

[8] M. Dreze, B. Charloteaux, S. Milstein, P.-O. Vidalain, M. a. Yildirim, Q. Zhong, N. Svrzikapa, V. Romero, G. Laloux, R. Brasseur, J. Vandenhaute, M. Boxem, M. E. Cusick, D. E. Hill, and M. Vidal. 'Edgetic' perturbation of a C. elegans BCL2 ortholog. *Nature methods*, 6(11):843–9, Nov. 2009.

[9] R. Durrett, J. Gleeson, A. L. Lloyd, P. J. Mucha, F. Shi, D. Sivakoff, J. E. S. Socolar, and C. Varghese. Graph fission in an evolving voter model. *Proceedings of the National Academy of Sciences*, 109(10):3682–7, 2012.

[10] P. Erds and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 1959.

[11] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129, Jan. 2009.

[12] C. E. Gounaris, K. Rajendran, I. G. Kevrekidis, and C. a. Floudas. Generation of networks with prescribed degree-dependent clustering. *Optimization Letters*, 5(3):435–451, Apr. 2011.

[13] C. E. Gounaris, K. Rajendran, I. G. Kevrekidis, and C. A. Floudas. Designing Networks : A Mixed-Integer Linear Optimization Approach. 2013.

[14] D. Haussler. Convolution Kernels on Discrete Structures. Jan. 1999.

[15] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 1999.

[16] V. Latora and M. Marchiori. Efficient Behavior of Small-World Networks. *Physical Review Letters*, 87(19):198701, Oct. 2001.

[17] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, Feb. 2004.

[18] B. Ráth. Time evolution of dense multigraph limits under edgeconservative preferential attachment dynamics. *Random Structures & Algorithms*, pages 1–29, 2012.

[19] J. W. Raymond, E. J. Gardiner, and P. Willett. Heuristics for similarity searching of chemical graphs using a maximum common edge subgraph algorithm. *Journal of Chemical Information and Computer Sciences*, 42(2):305–16, 2002.

[20] A. Robles-Kelly and E. R. Hancock. String Edit Distance, Random Walks and Graph Matching. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):315–327, May 2004.

[21] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.

[22] F. Shi, P. J. Mucha, R. Durrett, and C. Hill. A multi-opinion evolving voter model with infinitely many phase transitions. 2013.

[23] H. Yu and E. Hancock. String Kernels for Matching Seriated Graphs. *18th International Conference on Pattern Recognition*, pages 224–228, 2006.