CONTEST: A Controllable Test Matrix Toolbox for MATLAB

Alan Taylor and Desmond J. Higham*

May 20, 2008

Abstract

Large, sparse networks that describe complex interactions are a common feature across a number of disciplines, giving rise to many challenging matrix computational tasks. Several random graph models have been proposed that capture key properties of real-life networks. These models provide realistic, parametrized matrices for testing linear system and eigenvalue solvers. CONTEST (CONtrollable TEST matrices) is a random network toolbox for MATLAB that implements nine models. The models produce unweighted directed or undirected graphs; that is, symmetric or unsymmetric matrices with elements equal to zero or one. They have one or more parameters that affect features such as sparsity and characteristic pathlength and all can be of arbitrary dimension. Utility functions are supplied for rewiring, adding extra shortcuts and subsampling, in order to create further classes of networks. Other utilities convert the adjacency matrices into real-valued coefficient matrices for naturally arising computational tasks that reduce to sparse linear system and eigenvalue problems.

Keywords: clustering, matrix computation, preferential attachment, random graph, rewiring, sparse matrix, small-world.

1 Motivation

Networks describing connectivity structures arise across a vast range of application areas. Examples where it has proved useful to record data include interactions between genes [35], proteins [16], cortical regions [34, 59], internet nodes [21], web pages [11, 52], countries [20], co-authors [48], telephones [1], assets on the stock market [7] and members of various populations [13, 37, 51, 54, 64].

*Department of Mathematics, University of Strathclyde, Glasgow, G1 1XH, Scotland, UK (ta.atay@maths.strath.ac.uk and djh@maths.strath.ac.uk). DJH was supported by Engineering and Physical Sciences Research Council grants GR/S62383/01 and EP/E049370/1.

Typical data mining and visualisation tasks reduce to linear system or eigenvalue computations with the large, sparse adjacency matrices that define the interactions. Several random graph models, that is, formulas for probabilistically inserting connections, have been derived that attempt to capture the key topological properties of real-life networks. Important goals for such work are to understand how a network has reached its current state and to predict how it will evolve. From a numerical analysis perspective, these random graph models are an extremely useful source of realistic, controllable test matrices for linear algebra software. This provides the motivation for the MATLAB toolbox CONTEST (CONtrollable TEST matrices), which implements nine popular random network models, along with various utility functions for post-processing the networks. CONTEST is available from the website

`http://www.maths.strath.ac.uk/research/groups/numerical_analysis/contest`

The codes were developed and tested under MATLAB Version 7.4.0.287 (R2007a). As supplementary material at the website, we record performance results for MATLAB's built-in iterative linear system solvers `pcg`, `qmr`, `symmlq`, `lsqr`, `minres`, `cgs`, `gmres`, `bicg` and `bicgstab` using test matrices from the toolbox.

This article is arranged as follows. Section 2 gives a very brief overview of the historical development of random network models. In section 3 we describe each of the nine models and the corresponding MATLAB code. Section 4 introduces the utility functions for altering existing networks, setting up coefficient matrices arising in common tasks and checking some basic topological properties. In section 5 we give a very brief illustration of the toolbox in use, and we summarize the aims of this work in section 6.

Our notation is as follows. We let n denote the number of nodes in a network, with $a_{ij} = a_{ji} = 1$ if nodes i and j are connected and $a_{ij} = a_{ji} = 0$ otherwise. So the adjacency matrix $A \in \mathbb{R}^{n\times n}$ is symmetric. We always have $a_{ii} = 0$; so nodes cannot be self-connected. The *degree* of node i is found by counting its neighbours, $\deg_i := \sum_{j=1}^{n} a_{ij}$. For $\deg_i > 1$ the *curvature* or *clustering coefficient* of node i is found by counting how many pairs of these neighbours are themselves connected, and dividing this number by the maximum possible number of connections, $\deg_i(\deg_i - 1)/2$. A definition in terms of MATLAB comands is given in section 4.7.1.

A call to one of the random network functions in the toolbox will generate an $A \in \mathbb{R}^{n\times n}$ as an independent instance drawn from a random network model. The randomness is driven entirely by MATLAB's built in pseudo-random number generators, `rand` and `randn`, and our codes do not alter their states. So the user can get back the same matrix by re-setting the states of these two random number generators. For consistency, we always generate adjacency matrices with the `sparse` attribute, even though for some parameter values a full matrix may arise (for example, with the extreme choice of $p = 1$ in the Gilbert model of section 3.1).

Although we produce only symmetric adjacency matrices, it is straightforward to create unsymmetric versions, corresponding to directed networks, by combining the upper and lower triangles from two independent samples from the same model. For example, calling `A = erdrey(n,m)` and `B = erdrey(n,m)`, where `erdrey` described in section 3.1.1 implements the Erdős–Rényi model, we could set `C = triu(A) + tril(B)`.

2 Background

It has been repeatedly observed that real connectivity networks are neither completely regular lattices nor classical random graphs. Following the landmark paper of Watts and Strogatz [63], there has been a resurgence of interest in the idea of designing probabilistic models that capture important topological properties of real networks. Watts and Strogatz coined the phrase *small world network* to describe a regime where small pathlengths coexist with large clustering coefficients (nodes tend to live in cliquey, well-connected subgraphs and yet the network can be globally traversed with relatively few links). They also showed that this pair of properties arise when an appropriate amount of disorder is added to a regular lattice.

Another key property that is claimed to be common in real networks is a *scale-free degree distribution*,

$$\frac{\text{Number of nodes of degree } k}{n} \propto k^{-\gamma}, \tag{1}$$

where γ is a constant, typically in the range $2 \leq \gamma \leq 3$. The *preferential attachment* model of Barabási and Albert [5] attempts to describe the way a network might grow when new nodes are added and new connections formed, and it produces scale-free degree distributions. More recently, however, the prevalence of the scale-free property has been questioned, at least in the context of biological networks [36, 55, 60].

In addition to small worlds and scale-freeness, a third dominant concept is that of *motifs* [3, 45]. A motif is a subgraph that is significantly overrepresented (relative to the occurrence of that subgraph in a "randomized" version of the network). These motifs may be regarded as the basic building blocks of the networks, and hence understanding their roles gives valuable insights into how the overall network operates [41, 42]. The closely related idea of *graphlet frequency* was introduced in [55] as a means to compare networks and further developed in [56]. Two networks are "close" if they are made up of building blocks in the same relative proportions. This gives a powerful and comprehensive means to check whether a probabilistic model is capturing topological properties of real networks and to decide which models are most appropriate. Using these ideas, the software tool GraphCrunch for network comparison was developed in [43].

Overall, a recent and rapid expansion in theoretical and empirical research activity has produced several models for computing networks in a controlled manner that are "close" to real life networks in a well-defined sense. It is our tenet that these computable networks are therefore excellent candidates for test matrices.

Although well established sparse matrix test sets exist, [8, 15, 18], they have been built around fixed instances arising in particular application areas. Randomness is typically incorporated very simplistically. For example, Matrix Market [8] with website URL `http://math.nist.gov/MatrixMarket/` makes available the random generators `DLATMR/ZLATMR` from LAPACK [4], which independently assign random samples from a given distribution across the entries of an array and then randomly reset elements to zero in order to achieve a given level of sparsity. In [15], Davis argues that "random sparse matrices" are not appropriate for testing sparse matrix algorithms; however, those comments would appear to be aimed at different classes of matrices to those considered here. The models implemented in CONTEST use randomness to capture properties that are commonly observed in complex interaction networks.

The code in CONTEST was written to exploit vectorization and to use matrix-vector level operations where possible, but ultimately our priority was to allow sparse matrices of the largest possible dimension to be computed. A secondary aim was produce short, readable and maintainable programs. The importance of memory allocation and usage when generating sparse matrices in MATLAB is discussed in [23] and in NA Digest at `http://www.netlib.org/na-digest-html/07/v07n28.html#1`. Our justification for not focusing on execution time is that the tasks that will typically be performed with the matrices—eigensolves, linear systems solves, factorizations—will usually be more computationally expensive than the matrix generation phase.

3 Models

In this section, we give brief descriptions of the nine models implemented, and show how to use the corresponding MATLAB functions. In each case, the output argument, `A`, is a sparse, symmetric, zero-diagonal matrix of dimension `n`, with `n` being the first of the input arguments. The remaining input arguments take default values if not specified in the function call. Default parameters have been chosen to ensure that `A` corresponds to a connected (irreducible) graph with high probability, with the exception of `sticky` in section 3.7.1, which, by construction, may produce many small disconnected subgraphs. In Figure 1 we show a spy plot for each of the nine models using `n=100`; this dimension was chosen to make the visualisation clearer—in practice values of n of the order 10^4 or higher would be more realistic.

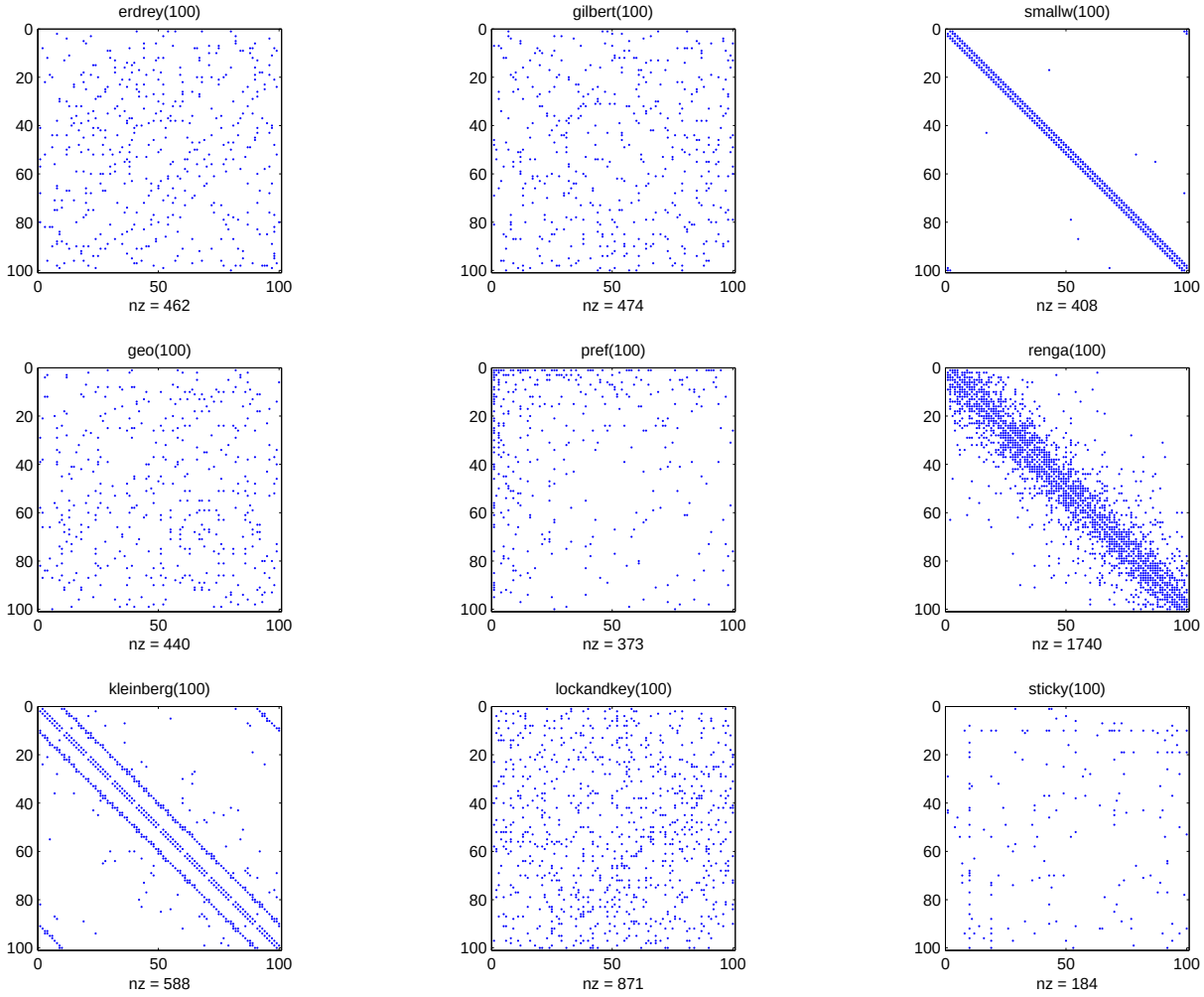

Figure 1: Spy plots showing nonzero patterns for a 100×100 sample from each of the nine models.

3.1 Classical

Random graph theory began in earnest in the late 1950s, with the two classical models in [22] and [19]. These models are usually referred to as $\mathcal{G}(n, p)$ and $\mathcal{G}(n, m)$, but to help distinguish between them we will use the names Gilbert and Erdős–Rényi.

In Gilbert's model [22] a fixed probability p is specified, and then each pair of nodes is, independently, connected with probability p. In the Erdős–Rényi model [19] the number, m, of edges in the network is specified. (Of course, m must

be no more than the maximum possible number of edges, $n(n-1)/2$.) We then select uniformly at random from the set of all graphs containing n nodes and m edges.

The properties of these classical random graphs have been well studied [2, 9], although in terms of currently adopted measures, such as pathlengths, clustering coefficients and graphlet frequencies, they cannot be regarded as accurate models of realistic networks [16, 55, 63]. Our implementation for the Gilbert class is taken from [6, Algorithm 1].

3.1.1 Classical Codes: `gilbert` and `erdrey`

The function `gilbert(n,p)` returns an instance from the Gilbert class. The optional second input argument defaults to $\log(n)/n$, so `A = gilbert(n)` is equivalent to `A = gilbert(n,log(n)/n)`. Similarly, `A = erdrey(n,m)` produces an Erdős–Rényi random graph, with m defaulting to the smallest integer bigger than $n\,\log(n)/2$.

3.2 Small World

Motivated by the "small world" concept of the experimental psychologist Stanley Milgram [44], Watts and Strogatz [63] proposed a random graph model that can be regarded as interpolating between a regular, periodic lattice and a classical random graph. Although the original work used re-wiring, it is now more common to introduce randomness via the addition of shortcuts [31, 49]. Hence, in our Watts-Strogatz model we begin with a k-nearest neighbour ring (nodes i and j are connected if and only if $|i-j| \leq k$ or $|n-|i-j|| \leq k$). Then, each node is considered independently in turn. With fixed probability p a node is given an extra link—a short cut—connecting it to a node chosen uniformly at random across the network. (At the end of this process, self links and repeated links between nodes are removed.)

3.2.1 Small World Code: `smallw`

The function `smallw` returns an instance of the Watts-Strogatz model, with syntax according to `A = smallw(n,k,p)`. The optional input arguments `k` and `p` default to 2 and 0.1, respectively. From a linear algebra perspective, the adjacency matrix has a symmetric, banded Toeplitz structure, with extra nonzeros added uniformly and symmetrically at random. We note that `smallw` makes use of the utility function `short` that is described in section 4.2.1.

3.3 Geometric

A two-dimensional, non-periodic, *geometric random graph* may be defined as follows. First, each of the n nodes is placed at random in the unit square—

more precisely, the ith node is given coordinates (x_i, y_i), where $\{x_i, y_i\}_{i=1}^n$ are independent and identically distributed with uniform (0,1) distribution. Next, for some specified radius, r, nodes i and j are connected if and only if $(x_i - x_j)^2 + (y_i - y_j)^2 \leq r^2$. In words, an edge denotes that two nodes were placed no more than Euclidean distance r apart. Figure 2 illustrates the process with $n = 100$ and $r = 0.2$.

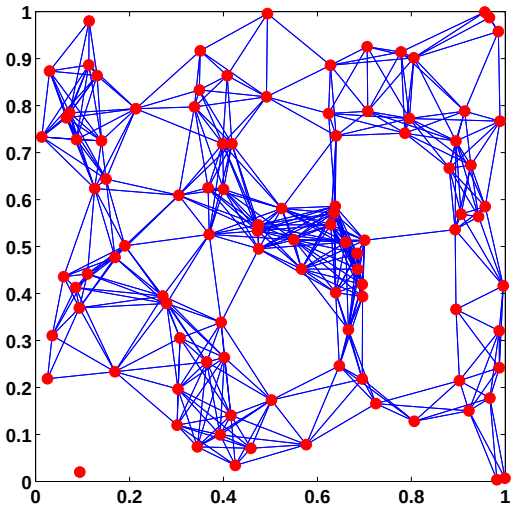

Figure 2: Construction of a geometric random graph. Here, $n = 100$ and $r = 0.2$.

We emphasize that the resulting graph is simply the usual list of nodes and edges. Information about the precise locations $\{x_i, y_i\}_{i=1}^n$ is not part of the final mathematical object. Natural generalizations are possible.

Dimension: the nodes can be randomly assigned to locations in the unit cube in $\mathbb{R}^m$, for some $m > 2$.

Periodicity: distance can be measured in a wrap-around fashion, so that, for example, in the unit square, $(x_i - x_j)^2 + (y_i - y_j)^2$ is replaced by

$$\left(\min\left(|x_i - x_j|, 1 - |x_i - x_j|\right)\right)^2 + \left(\min\left(|y_i - y_j|, 1 - |y_i - y_j|\right)\right)^2.$$

Norm: the Euclidean norm can be replaced by any other vector norm.

Much theory is available concerning properties of geometric random graphs; see [53] for a comprehensive treatment. Recently Pržulj et al. [55] showed that two and three dimensional non-periodic versions, using the Euclidean norm, give surprisingly accurate reproductions of many features of real biological networks and an algorithm that tests for geometric structure is developed in [32].

3.3.1 Geometric Code: `geo`

The call `A = geo(n,r,m,per,pnorm)` returns an instance of a geometric random graph. There are four optional input arguments:

- `r` specifies the radius, defaulting to $\sqrt{1.44/n}$, which is motivated by the asypmtotic ($n \to \infty$) level that guarantees connectivity in two dimensions [53].
- `m` specifies the dimension, defaulting to 2,
- `per` is a logical variable specifying whether periodic distance is to be used, defaulting to `per = 0`; not periodic,
- `pnorm` specifies the L_p-norm to be used, defaulting to 2.

3.4 Preferential Attachment

Barabási and Albert [5] used the concept of preferential attachment to develop random graphs with scale-free degree distributions. In this model, the network grows—new nodes are added and linked in to the existing network—until n nodes have been created. For some fixed integer $d \geq 1$, each new node is given d links on arrival. These new connections are not chosen unformly; the new node links to an existing node with a probability that is proportional to the current degree of that node. In this way, well-connected nodes tend to become even better connected (the rich get richer) as the network evolves. Our precise model is a translation into MATLAB of [6, Algorithm 5], which uses the specification in [10].

3.4.1 Preferential Attachment Code: `pref`

The call `A = pref(n,d)` returns an instance of a preferential attachment graph, using a single node as the initial network. The degree parameter `d` defaults to 2.

3.5 Range Dependent

3.5.1 RENGA

Yeast two hybrid protein-protein interaction (PPI) networks have proteins as nodes. Two nodes share an undirected edge if they have been experimentally observed to interact [65]. Motivated by the structure of PPI networks, Grindrod [25] proposed and analyzed a random graph model that, in a sense, generalizes Watts–Strogatz. In this model, the nodes have a natural linear ordering, $i = 1, 2, \ldots, n$. Independently over all pairs of nodes, we then insert a link between nodes i and j with probability $\alpha\lambda^{|j-i|-1}$, where $\alpha > 0$ and $\lambda \in (0, 1)$ are fixed parameters. The choice $\alpha = 1$ ensures that adjacently ordered nodes are always connected.

The geometric factor $\lambda^{|j-i|-1}$ causes long-range edges to be less common than short-range edges.

Further analysis and generalizations of this model, now refered to as RENGA, appear in [26, 29, 30]. Closely related models have also been used in percolation theory [24].

3.5.2 RENGA Code: `renga`

The call `A = renga(n,lambda,alpha)` returns an instance of a RENGA, with `lambda` defaulting to 0.9 and `alpha` defaulting to 1.

3.5.3 Kleinberg

Kleinberg [38] defined a variation of the Watts-Strogatz model, and used it to examine which types of navigation algorithm can exploit the existence of short cuts. Kleinberg's model is based on a periodic, two-dimensional lattice: the $n = m^2$ nodes can be thought of as being equally spaced throughout a square, with each node having a location of the form $(i, j) \in \mathbb{R}^2$ where the integers i and j run from 1 to m. Every node is given short range connections to its neighbours that are a lattice (Manhattan) distance of at most p away. Then each node is given q further 'long-range' connections. For a given node, u, the recipient, v, of each such long-range connection is chosen independently at random, with probability proportional to $r^{-\alpha}$. Here, r is the lattice distance between u and v and $\alpha \geq 0$ is a fixed parameter.

3.5.4 Kleinberg Code: `klein`

The call `A = kleinberg(n,p,q,alpha)` generates an instance of the Kleinberg model. If the input dimension, `n`, is not a perfect square then the output matrix has dimension `(round(sqrt(n)))^2`. Default values are `p = 1`, `q = 1` and `alpha = 2`.

3.6 Lock and Key

Using some basic biological insights, Thomas et al. [61] proposed a class of random graphs that model PPI networks. This class of models was further analysed in [47], where it was used to extract new biological information from real PPI data sets. The underlying modeling idea is that two proteins interact because they share physically matching parts, which, following [47], we refer to as *locks* and *keys*. There will be several different types of key, which we can think of as labeled by colors (red, green, blue, etc.) and for each type of key there is a matching lock (red, green, blue, etc.). In the model, each protein has the same chance of possessing each color of lock and each color of key. More precisely, for a given number of colors, m, we take each node in turn and independently assign it each

possible lock and key with some fixed probability p. The graph is then generated according to the rule that two nodes share an edge if and only if one possesses a key and the other possesses a lock of the same color. Self links are removed.

3.6.1 Lock and Key Code: `lockandkey`

The call `A = lockandkey(n,m,p)` returns an instance of a lock and key graph where there are `m` different lock and key colors and each type of lock and key is handed out independently with fixed probability `p`. Default values are `m = ceil(n*log(n))` and `p = 1/n`.

3.7 Stickiness

The stickiness model was introduced in [57] to model PPI networks. It was motivated as a simplified version of the lock and key framework in which parameters could be fitted to real data. Here, a nonnegative vector $\widehat{d} \in \mathbb{R}^n$ is given, representing the scaled degree distribution of some target network; more precisely, $\widehat{d}_i = \deg_i / \sqrt{\sum_{j=1}^{n} \deg_i}$, where $\deg_i$ is the degree of the ith node in the target. Then a new random network is produced by connecting nodes i and j with probability $\widehat{d}_i \widehat{d}_j$. In this way the *expected degrees* in the random model match the target degrees. This model was found to be more accurate than previously proposed models at reproducing topological properties of PPI networks.

3.7.1 Stickiness Code: `sticky`

The call `A = sticky(deg)` generates an instance of a stickiness graph with expected degree distribution given by the one-dimensional array `deg`. To be consistent with our general philosophy that all models can be called with a single input argument, `n`, representing the dimension, we allow an exception where `sticky` is called as `A = sticky(n)`, with `n` a positive integer. In this case `A` will be an instance of a stickiness graph of dimension `n` with a scale-free expected degree distribution of the form (1) with $\gamma = 2.5$. It is also possible to specify two input parameters, a call `A = sticky(n,gamma)` specifies the value of γ to be used in (1).

4 Utility Functions

4.1 Rewiring

The Watts-Strogatz model [63] added randomness to a ring network by *rewiring* some edges. For a general undirected network, we define a rewiring process as follows, in terms of a fixed parameter p. Each entry in the lower triangle of the original adjacency matrix is examined in turn. If $a_{ij} \neq 0$ then, independently

with probability p, we reset $a_{ij} = a_{ji} = 0$, choose a node k uniformly at random from all non-neighbours of node i, and set $a_{ik} = a_{ki} = 1$.

4.1.1 Rewiring Code: `rewire`

The call `R = rewire(A,p)` takes an adjacency matrix `A` and returns a rewired adjacency matrix `R`. The rewiring probability `p` defaults to `p = log(n)/n`.

4.2 Shortcuts

Rewiring has the theoretical drawback that it may cause a connected network to become unconnected. Adding *shortcuts* is an alternative procedure that gives very similar topological effects [49] but does not degrade connectivity. In this case the parameter p is a fixed a probability that is used independently over all nodes. For each node, with probability p we add a new link from that node to a node chosen uniformly at random across the whole network. Self links are then removed and repeated links treated as single links.

4.2.1 Shortcut Code: `short`

The call `S = short(A,p)` takes an adjacency matrix `A`, adds shortcuts and returns the new adjacency matrix `S`. The shortcut probability `p` defaults to `log(n)/n`.

4.3 Subsampling

Information is often missing from real life connectivity data sets [17]. These omissions may be caused, for example, by errors in experimental observations (false negatives) or by an inherent restriction on the number or type of observations that can be made. In the case of yeast two hybrid PPI networks, it is widely accepted that the reported network is merely a noisy subset of the underlying "true" network, and we can think of the given network as being generated from a "subsampling" operation on the larger version [62]. Interestingly, it has been discovered that the subsampling operation may dramatically alter the topological properties of a network [17, 27, 58].

We have implemented two subsampling algorithms. Given the adjacency matrix for a network they return the adjacency matrix for a network consisting of a subset of those nodes and edges. The first algorithm does an unbiased, uniform node removal involving a fixed parameter p. Each node is considered in turn, and with independent probability $1-p$ we remove that node and all edges that involve it, that is, we delete that row and column from the adjacency matrix. The second algorithm uses a bait and prey approach, along the lines of [27], which models the generation of certain PPI data sets. Here, we use two fixed parameters, `bait` and `prey`. A proportion `bait` of the nodes are chosen as baits. Then, for each

bait, a proportion `prey` of its edges are recorded, along with the prey nodes that are linked to the bait by those edges. The final subsampled network consists of the bait-prey edges and all the nodes that they involve.

4.3.1 Supsampling Codes: `unisample` and `baitsample`

The call `U = unisample(A,p)` takes an adjacency matrix `A` and returns a subnetwork `U` formed from an unbiased, uniform node removal. The probability `p` defaults to `0.5`.

The bait and prey algorithm can be called as `B = baitsample(A,bait,prey)`, with defaults `bait = 0.5` and `prey = 0.5`.

4.4 Laplacian Matrices

An undirected network can be characterised by its adjacency matrix, and basic linear algebra tells us that the eigenvectors and eigenvalues of this matrix carry relevant information. However, spectral graph theory [12] has shown that it is generally more useful to look at the spectrum of the so-called Laplacian. There are two different matrices that take this name in the literature. We distinguish between them as follows.

- The *graph Laplacian* has the form $D - A$.
- The *normalized graph Laplacian* has the form $\widehat{D}^{-\frac{1}{2}}(D - A)\widehat{D}^{-\frac{1}{2}}$.

Here $D = \text{diag}(\deg_i)$ and $\widehat{D} = D$, with the exception that we take $\widehat{D}_{ii} = 1$ in the case where $\deg_i = 0$,

Clustering and partitioning tasks can be tackled by computing eigenvectors corresponding to small eigenvalues of these matrices. In particular the *Fiedler vector* and *normalised Fiedler vector* of a connected network are defined to be the eigenvectors corresponding to the second smallest eigenvalues of the Laplacian and normalised Laplacian, respectively. Specific software exists for computing this type of information [14, 28, 33].

4.4.1 Laplacian Matrix Codes: `lap`

The call `L = lap(A,nl)` takes a symmetric adjacency matrix `A` and returns a Laplacian; `nl=0` for unnormalized and `nl=1` for normalized. The default is `nl=1`.

4.5 PageRank matrix

The PageRank algorithm returns a vector whose ith entry indicates the "importance" of the ith node in a network. The algorithm was invented by Page and Brin and forms the heart of the search engine Google [39, 52]. PageRank was

originally designed for the directed network where nodes are web pages and edges are hypertext links, but it has also been used on networks in biology [46]. Given an adjacency matrix A, the Pagerank vector x solves the linear system

$$Px = \mathbf{1}, \qquad \text{where } P = I - dA^T \widehat{D}^{-1}. \tag{2}$$

Here, $d \in (0, 1)$ is a scalar parameter, the diagonal degree matrix $\widehat{D}$ is defined in section 4.4 and $\mathbf{1}$ denotes the vectors of 1s. More precisely, when A is unsymmetric we consider the *out degree*, so $D = \text{diag}\left(\sum_{j=1}^{N} a_{ij}\right)$ and $\widehat{D} = \text{diag}\left(\max(D_{ii}, 1)\right)$.

4.5.1 PageRank Code: `pagerank`

The call `P = pagerank(A,d)` takes an adjacency matrix `A` and returns the PageRank matrix `P`, with `d` defaulting to 0.85. The matrix `A` is not assumed to be symmetric—directed edges are allowed.

4.6 Mean Hitting Time Matrix

In many applications it is useful to consider the discrete time, finite state space, Markov chain that arises naturally from a network [40]. Here, if we are currently at node i then at the next time level we move to a node chosen uniformly among the neighbours of node i. The *transition matrix* for this Markov chain thus has the form $D^{-1}A$. Fixing a node, i, the the *mean hitting time* for node j is defined to be the average number of steps required for the Markov chain to reach state j, given that it starts at state i. The vector of mean hitting times can be found by solving the linear system $Mx = \mathbf{1}$, where $M \in \mathbb{R}^{n-1 \times n-1}$ is the transition matrix with its ith row and column removed [50].

4.6.1 Mean Hitting Time Code: `mht`

The call `M = mht(A,i)` takes an adjacency matrix `A` with nonzero out degrees and returns the mean hitting time matrix `M` for a chain that starts at node `i`, with `i` defaulting to `1`. The matrix `A` is not required to be symmetric.

4.7 Pathlength and Curvature

The *pathlength* between nodes i and j is the smallest number of edges that must be crossed to reach j starting from i. In terms of the adjacency matrix, A, the pathlength between nodes i and j can be characterised as the smallest integer $k \geq 1$ such that $(A^k)_{ij} \neq 0$. If $(A^{n-1})_{ij} = 0$ then there is no suitable path and the pathlength may be regarded as infinite.

The *curvature*, or *clustering coefficient*, of a node was defined in section 1. In MATLAB notation, the vector of clustering coefficients may be computed as

```
diag(A^3)/(sum(A).*(sum(A) - 1))
```

4.7.1 Pathlength and Curvature Codes: `pathlength` and `curvature`

The call `Path = pathlength(A)` returns an array `Path` of the same dimension as the adjacency matrix `A`, such that `Path(i,j)` is the pathlength from node `i` to node `j`. We always set `Path(i,i)=0` and we use `Path(i,j)=inf` to denote that no path exists.

The call `curv = curvature(A)` takes an adjacency matrix `A` of dimension `n` and returns a one-dimensional array `curv` of length `n`, such that `curv(i)` records the curvature of node `i`. A second input argument is allowed. The call `curv = curvature(A,ind)` returns the maximum curvature if `ind` is the string `'max'`, the average curvature if `ind` is the string `'ave'` and the curvature for the `i`th node if `ind` is the integer `i`. Undefined curvature evaluates to `NaN`.

5 Computational Experiment

For a brief illustration of the toolbox in use, we follow Davis [15] by examining the complexity of the minimum degree ordering algorithm, as implemented in MATLAB's `amd`. Letting L denote the Cholesky factor of the appropriate permuted version of A, we plot the run time, scaled by $|L|$, against $|L|$, on a log-log scale. Davis [15] distinguished between matrices from a deterministic test set coming from problems with and without inherent geometry. To mirror this, Figure 3 shows results for matrices arising from the Gilbert class, using `gilbert`, where there is no inherent structure, and Figure 4 shows results for matrices arising from the Kleinberg class, using `klein`, where there is an underlying lattice. The least-squares slope is indicated by a solid line. In each case the matrix dimension n was varied between 50 and 10,000. The test programs are available from the testing section of the toolbox website. The figures are consistent with the rule of thumb mentioned in [15] that the run time is typically below $O(|L|)$.

6 Summary: Networks as Test Matrices

The motivation for this work is that recent random network models make excellent candidates for sparse test matrices. The models capture features of interaction data observed across a wide range of application areas, and they incorporate parameters that allow the user to control topological features, including sparsity and the distribution of degree and clustering coefficients. Naturally arising computational tasks in network science present challenging test problems for general (symmetric and unsymmetric) linear system solvers and symmetric eigenvalue routines.

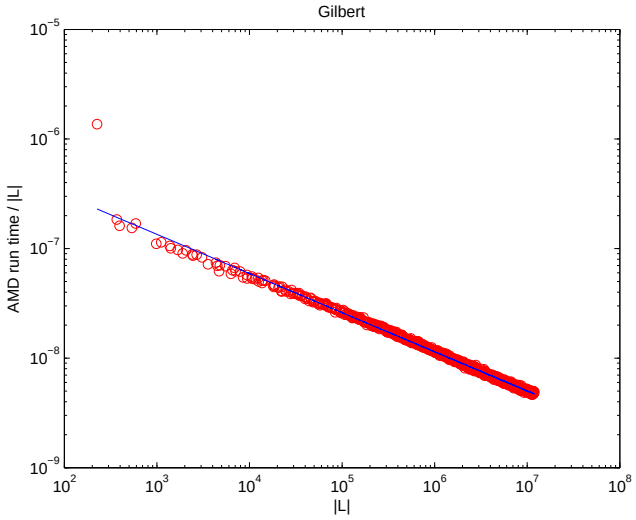

Figure 3: `amd` run times for Gilbert model

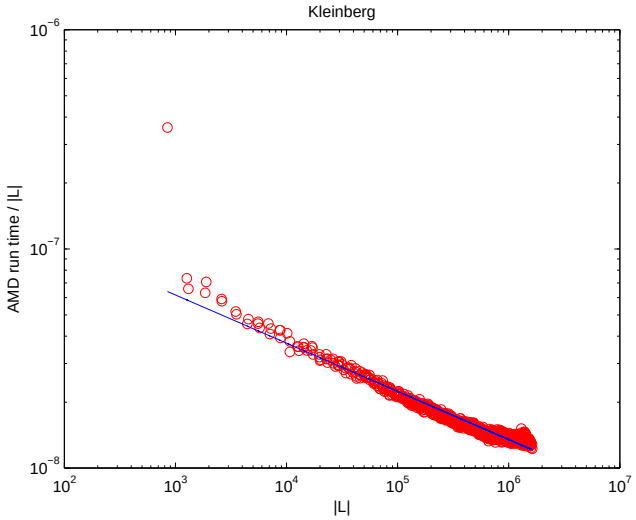

Figure 4: `amd` run times for Kleinberg model

References

[1] J. Abello, A. Buchsbaum, and J. Westbrook, *A functional approach to external graph algorithms*, Lecture Notes in Computer Science, 1461 (1998), pp. 332–343.

[2] R. Albert and A.-L. Barabási, *Statistical mechanics of complex networks*, Reviews of Modern Physics, 74 (2002), pp. 47–97.

[3] U. Alon, *An Introduction to Systems Biology*, Chapman & Hall/CRC, London, 2006.

[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, SIAM, PA, third ed., 1999.

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

[6] V. Batagelj and U. Brandes, *Efficient generation of large random networks*, Phys. Rev. E, 71 (2005), p. 036113.

[7] V. Boginski, S. Butenko, and P. M. Pardalos, *On structural properties of the market graph*, in Innovations in Financial and Economic Networks, A. Nagurney, ed., Edward Elgar Publishers, 2003, pp. 29–45.

[8] R. Boisvert, R. Pozo, K. Remington, R. Barrett, and J. Dongarra, *Matrix Market: a web resource for test matrix collections*, in The Quality of Numerical Software: Assessment and Enhancement, R. Boisvert, ed., London, 1997, Chapman and Hall, pp. 125–137.

[9] B. Bollobás, *Random Graphs*, Academic, London, 1985.

[10] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády, *The degree sequence of a scale-free random graph process*, Random Structures and Algorithms, 18 (2001), pp. 279–290.

[11] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, *Graph structure of the web*, Computer Networks, 33 (2000), pp. 309–320.

[12] F. Chung, *Spectral Graph Theory*, American Mathematical Society, Providence, RI, 1997.

[13] M. J. Conyon and M. R. Muldoon, *The small world of corporate boards*, Journal of Business Finance and Accounting, 33 (2006), pp. 1321–1343.

[14] T. Cour, F. Benezit, and J. Shi, *Spectral segmentation with multiscale graph decomposition*, in IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, 2005, pp. 1124–1131.

[15] T. Davis, *The University of Florida sparse matrix collection*, Tech. Rep. CISE Department, REP-2007-298, University of Florida, USA, 2007.

[16] E. de Silva and M. Stumpf, *Complex networks and simple models in biology*, J. R. Soc. Interface, 2 (2005), pp. 419–430.

[17] E. de Silva, T. Thorne, P. Ingram, I. Agrafiot, J. Swire, C. Wiuf, and M. P. H. Stumpf, *The effects of incomplete protein interaction data on structural and evolutionary inferences*, BMC Biology, 4:39 (2006).

[18] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Soft., 15 (1989), pp. 1–14.

[19] P. Erdös and A. Rényi, *On random graphs*, Publ. Math. Debrecen, 6 (1959), pp. 290–297.

[20] G. Fagiolo, *Clustering in complex directed networks*, Physical Review, 76 (2007), p. 026107.

[21] M. Faloutsos, P. Faloutsos, and C. Faloutsos, *On power-law relationships of the internet topology*, Computer Communications Review, 29 (1999), pp. 251–262.

[22] E. N. Gilbert, *Random graphs*, Ann. Math. Statist., 30 (1959), pp. 1141–1144.

[23] J. R. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in MATLAB: design and implementation*, SIAM J. Matrix Analysis Applications, 13 (1992), pp. 333–356.

[24] G. Grimmett, *Percolation*, Springer, 2nd ed., 1999.

[25] P. Grindrod, *Range-dependent random graphs and their application to modeling large smal-world proteome datasets*, Phys. Rev. E, 66 (2002), p. 066702.

[26] P. Grindrod, D. J. Higham, and G. Kalna, *Periodic reordering*, Tech. Rep. 6, University of Strathclyde, Department of Mathematics, 2008.

[27] J. D. H. Han, D. Dupuy, N. Bertin, M. E. Cusick, and V. M., *Effect of sampling on topology predictions of protein-protein interaction networks*, Nature Biotechnology, 23 (2005), pp. 839–844.

[28] B. Hendrickson and R. Leland, *The Chaco user's guide: Version 2.0*, Tech. Rep. SAND94–2692, Sandia National Laboratories, Albuquerque, 1994.

[29] D. J. Higham, *Unravelling small world networks*, J. Comp. Appl. Math., 158 (2003), pp. 61–74.

[30] ——, *Spectral reordering of a range-dependent weighted random graph*, IMA J. Numer. Anal., 25 (2005), pp. 443–457.

[31] D. J. Higham and N. J. Higham, *MATLAB Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[32] D. J. Higham, N. Pržulj, and M. Rašajski, *Fitting a geometric graph to a protein-protein interaction network*, Bioinformatics, 24 (2008), pp. 1093–1099.

[33] Y. Hu and J. A. Scott, `HSL_MC73`*: A fast multilevel Fiedler and profile reduction code*, RAL-TR-2003-36, Numerical Analysis Group, Computational Science and Engineering Department, Rutherford Appleton Laboratory, 2003.

[34] L. Kamper, A. Bozkurt, K. Rybacki, A. Geissler, I. Gerken, K. E. Stephan, and R. Kötter, *An introduction to CoCoMac-Online. The online-interface of the primate connectivity database CoCoMac*, in Neuroscience Databases A Practical Guide, R. Kötter, ed., Norwell, MA, 2002, Kluwer Academic, pp. 155–169.

[35] S. A. Kauffman, *Metabolic stability and epigenesis in randomly constructed genetic nets*, J. of Theor. Biol., 22 (1969), pp. 437–467.

[36] R. Khanin and E. Wit, *How scale-free are gene networks?*, Journal of Computational Biology, 13 (2006), pp. 810–818.

[37] I. Z. Kiss, D. M. Green, and R. R. Kao, *The network of sheep movements within Great Britain: network properties and their implications for infectious disease spread*, J. Roy. Soc. Interface, 3 (2006), pp. 669–677.

[38] J. M. Kleinberg, *Navigation in a small world*, Nature, 406 (2000), p. 845.

[39] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, Princeton, 2006.

[40] L. Lovász, *Random walks on graphs: A survey*, in Paul Erdös is Eighty, D. Miklós, V. T. Sós, and T. Szönyi, eds., Budapest, 1996, János Bolyai Mathematical Society, pp. 353–398.

[41] S. Mangan and U. Alon, *Structure and function of the feed-forward loop network motif*, Proc. Nat. Acad. Sci., 100 (2003), pp. 11980–11985.

[42] S. Mangan, A. Zaslaver, and U. Alon, *The coherent feedforward loop serves as a sign-sensitive delay element in transcription networks*, J. Math. Biol., 334/2 (2003), pp. 197–204.

[43] T. Milenkovic, J. Lai, and N. Przulj, *GraphCrunch: A tool for large network analyses*, BMC Bioinformatics, 9:70 (2008).

[44] S. Milgram, *The small world problem*, Psychology Today, 2 (1967), pp. 60–67.

[45] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, *Superfamilies of evolved and designed networks*, Science, 303 (2004), pp. 1538–1542.

[46] J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert, *Generank: Using search engine technology for the analysis of microarray experiments*, BMC Bioinformatics, 6:233 (2005).

[47] ——, *A lock-and-key model for protein-protein interactions*, Bioinformatics, 2 (2006), pp. 2012–2019.

[48] M. E. J. Newman, *Who is the best connected scientist? a study of scientific coauthorship networks*, in Complex Networks, E. Ben-Naim, H. Frauenfelder, and Z. Toroczkai, eds., Springer, 2004, pp. 337–370.

[49] M. E. J. Newman, C. Moore, and D. J. Watts, *Mean-field solution of the small-world network model*, Phys. Rev. Lett., 84 (2000), pp. 3201–3204.

[50] J. R. Norris, *Markov Chains*, Cambridge University Press, 1997.

[51] R. N. Onody and P. A. de Castro, *Complex network study of Brazilian soccer players*, Phys. Rev. E, 70 (2004).

[52] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank citation ranking: Bringing order to the web*, tech. rep., Stanford Digital Library Technologies Project, 1998.

[53] M. Penrose, *Geometric Random Graphs*, Oxford Univeristy Press, 2003.

[54] M. A. Porter, P. J. Mucha, M. E. J. Newman, and C. M. Warmbrand, *A network analysis of committees in the United States House of Representatives*, Proc. Nat. Acad. Sci., 102 (2005), pp. 7057–7062.

[55] N. Pržulj, D. G. Corneil, and I. Jurisica, *Modeling interactome: Scale-free or geometric?*, Bioinformatics, 20 (2004), pp. 3508–3515.

[56] ——, *Efficient estimation of graphlet frequency distributions in protein-protein interaction networks*, Bioinformatics, 22 (2006), pp. 974–980.

[57] N. Pržulj and D. J. Higham, *Modelling protein-protein interaction networks via a stickiness index*, J. Roy. Soc. Interface, 3 (2006), pp. 711–716.

[58] M. Salathé, R. M. May, and S. Bonhoeffer, *The evolution of network topology by selective removal*, J. R. Soc. Interface, 2 (2005), pp. 533–536.

[59] O. Sporns and J. D. Zwi, *The small world of the cerebral cortex*, Neuroinformatics, 2 (2004), pp. 145–162.

[60] M. P. H. Stumpf, C. Wiuf, and R. M. May, *Subnets of scale-free networks are not scale-free: Sampling properties of networks*, Proc. Nat. Acad. Sci., 102 (2005), pp. 4221–4224.

[61] A. Thomas, R. Cannings, N. A. M. Monk, and C. Cannings, *On the structure of protein-protein interaction networks*, Biochemical Soc. Tranl., 31 (2003), pp. 1491–1496.

[62] B. Titz, M. Schlesner, and P. Uetz, *What do we learn from high-throughput protein interaction data?*, Expert Review of Proteomics, 1 (2004), pp. 111–121.

[63] D. J. Watts and S. H. Strogatz, *Collective dynamics of 'small-world' networks*, Nature, 393 (1998), pp. 440–442.

[64] R. J. Williams, E. L. Berlow, J. A. Dunne, A.-L. Barabási, and N. D. Martinez, *Two degrees of separation in complex food webs*, Proc. Nat. Acad. Sci., 99 (2002), pp. 12913–6.

[65] I. Xenarios, L. Salwinski, X. J. Duan, P. Higney, S. M. Kim, and E. D., *DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions*, Nucleic Acids Research, 30 (2002), pp. 303–305.