Publication 3

Sami Hanhijärvi, Gemma C. Garriga, and Kai Puolamäki. 2009. Randomization techniques for graphs. In: Proceedings of the 9th SIAM International Conference on Data Mining (SDM 2009). Sparks, Nevada, USA. 30 April - 2 May 2009. Society for Industrial and Applied Mathematics. Pages 780-791.

Randomization Techniques for Graphs

Sami Hanhijärvi, Gemma C. Garriga, Kai Puolamäki
Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Helsinki University of Technology, Finland
{firstname.lastname}@tkk.fi

Abstract

Mining graph data is an active research area. Several data mining methods and algorithms have been proposed to identify structures from graphs; still, the evaluation of those results is lacking. Within the framework of statistical hypothesis testing, we focus in this paper on randomization techniques for unweighted undirected graphs. Randomization is an important approach to assess the statistical significance of data mining results. Given an input graph, our randomization method will sample data from the class of graphs that share certain structural properties with the input graph. Here we describe three alternative algorithms based on local edge swapping and Metropolis sampling. We test our framework with various graph data sets and mining algorithms for two applications, namely graph clustering and frequent subgraph mining.

1 Introduction

Graph data arises in a wide variety of disciplines. For example, social networks consist of individuals and the social interactions between them. Another popular example is the World Wide Web, consisting of pages and their links, or very similarly, the Internet with its computer networks formed by routers and the physical connections between them. Further than this, input data in many application areas can potentially be cast into this relational format. Sometimes it is simpler to consider sets of graphs instead of a very large single graph. This is the case in chemo-informatics applications, where molecules can be represented by small graphs. Similar examples occur in biology and ecology, among many other fields from physical and life sciences. In all, graphs are remarkably versatile structures.

Studying the patterns and properties of graph data has become very important in many application areas. This situation has encouraged the data mining community to invest considerable research effort into uncovering interesting structures from graphs: subgraph mining [13, 30], clustering nodes into communities [26], link mining [10], virus propagation [29], just to cite some. An important question arising from this overload of graph mining results remains still largely ignored: how significant are the results found from the graph data? Currently, the results are mostly justified by the optimal or near optimal value of the defined objective function. For example this is the case in clustering algorithms. Yet, is the number of clusters found in the data significant? Or do those clusters indeed exist in the structure of the graph?

In traditional statistics the issue of significance testing has been thoroughly studied for many years. Given the observed data and a structural measure calculated from it, a hypothesis testing method computes the probability that the observed data was drawn from a given null hypothesis. Randomization approaches produce multiple random datasets according to a null hypothesis about the data. If the computed structural measure of the original data deviates significantly from the measurements on the random datasets, then we can discard the null hypothesis and consider the result significant.

The main goal of this paper is to study randomization techniques on graphs. The key idea is to apply enough distortion to the original graph data to obtain random samples from a desired null distribution. We propose to use the null hypothesis that the structure in the original graph is explained by the graph statistics used to define the null distribution. These graph statistics can be, for example, the degree distribution of the nodes or the average clustering coefficient of the graph. Broadly, the results of the data mining algorithm from the input graph will be considered interesting if they are unlikely to be observed in the random graphs that have, approximately, the same values of the graph statistics that the input graph had. Here we focus on undirected unweighted graphs, and consider the specific applications of subgraph pattern mining and graph clustering.

The problem of generating graph samples from a

user-specified null distribution is a generalization of the swap randomization task [11]. The randomization framework we propose on graphs is based on three types of local edge swappings and Metropolis sampling. We show, for all three local transformations, how to preserve the degree distribution of the graph, and additionally, other user-specified graph statistics, such as the characteristic path length or the average clustering coefficient. The combinations of graph statistics to be preserved in the samples together with the type of edge swaps performed define the level of constraint in generating the random samples. Indeed, these constraints identify exactly the null hypothesis to answer the following in a statistical test: do the observed data mining results on the original graph convey any information in addition to the specified graph statistics? We show by experiments that the randomization techniques on graphs are useful and necessary in practice and show how to derive the empirical p-value of the data mining results on graphs.

The paper is organized as follows. Section 2 discusses related work. Section 3 introduces the preliminaries and notation, while Section 4 specifies exactly the graph randomization problem approached in this paper. Algorithms to solve the problem are shown in Section 5, where we discuss their specifics as well other technical details of our Metropolis sampling technique. Finally, in Section 6 and Section 7 we present the applications and experiments. We go through a final discussion and conclusions in Section 8.

2 Related work

Randomization tests, also called *permutation tests*, are a common method for statistical significance testing [9]. They are often used in, for example, biology [20] and ecology [18], to generate the distribution for the test statistic when no standard distribution, such as normal or χ^2, can be applied. The technique has also been previously applied to data mining [11], where a 0–1 matrix is randomized to assess the results of methods like frequent itemset discovery or clustering. The key idea in the approach is to swap 0's with 1's while maintaining the marginal distributions of rows and columns. Recently, the method has been extended to real valued matrices [25].

Also bioinformatics makes use of constructive graph models to define p-values for graphs [16]. Similar works contribute with Monte Carlo swapping methods in order to sample graphs and define empirical probabilities [27]. Our work extends those methods by different swapping algorithms that can efficiently preserve several user-defined graph statistics, not only the degree distribution. The specification of the null hypothesis is user dependent and hence, more flexible. This flexibility allows for more versatile statistical studies.

Privacy preservation is one application for randomization on graphs. The goal in a recent contribution [33] is to perform a few natural edge swap perturbations that will preserve the spectrum of the original graph, while protecting the identity privacy of the nodes in the network. Our contribution is similar to [33] in that we aim at preserving different statistics with the edge swap perturbations; from this perspective, the spectrum could be simply one of these statistics we wish to preserve. On the other hand, our final goal for randomizing is quite different from [33]: we eventually want to evaluate the significance of the data mining algorithms in a graph, and therefore, we need to ensure that the final randomized samples are taken uniformly at random from the set of all graphs with the same statistics as the original data. Performing only a few swaps as in [33] is not always enough, so there is a need to study the convergence of the randomization process.

3 Preliminaries

We consider unweighted undirected graphs $G = (V, E)$ defined by a set of n nodes V and a set of m edges E. Each edge is an undirected pair (u, v) connecting two nodes $u, v \in V$. For a node $v \in V$, we use $\Gamma(v)$ to denote the set of neighboring nodes, and $\delta(v) = |\Gamma(v)|$ to denote the degree of v, that is, the number of nodes adjacent to v. For a graph G, we will often denote the set of vertexes by $V(G)$ and the set of edges by $E(G)$.

The structure of a graph can be described by various *graph statistics*. From several of them in the literature (see [19, 23]), we will use the degree distribution, average clustering coefficient and characteristic path length in this paper. These graph statistics summarize compactly the structural relations between the nodes in the graph; however, the methods presented here are general enough and can be extended to any number of other graph statistics.

The clustering coefficient of a node $v \in V(G)$ is the fraction of links the neighbors of the node have among them with respect to all possible such links,

$$\text{CC}(v) = \frac{|\{(u,w)|u,w \in \Gamma(v) \text{ and } (u,w) \in E\}|}{|\Gamma(v)|(|\Gamma(v)|-1)/2}. \tag{3.1}$$

The average clustering coefficient reads,

$$\text{AvgCC}(G) = \frac{1}{|V(G)|} \sum_{v \in V(G)} \text{CC}(v). \tag{3.2}$$

The characteristic path length of G is calculated as

the mean of all pairs shortest paths between the nodes,

$$\text{CPL}(G) = \frac{1}{|V(G)|^2} \sum_{u,v \in V(G)} d_G(u,v), \tag{3.3}$$

where $d_G(u,v)$ is the shortest graph theoretic distance from u to v in G. To generalize the choice of the graph statistic, we will denote it generally as $r(G) \in \mathbb{R}$, or simply r.

Our contribution is set within the framework of statistical hypothesis testing. We denote a *test statistic* as $t(\mathcal{A}(G)) \in \mathbb{R}$, or t for short, where $\mathcal{A}$ is a data mining algorithm and $\mathcal{A}(G)$ is the result of the algorithm on the graph G. The test statistic could be defined, for example, to be the value of the objective function: the value of the minimum cut in graph clustering or the number of patterns in frequent graph mining. However in principle, it can be any function from the space of results to a real number. Note the difference between the graph statistic r and the test statistic t. The former strictly measures an aspect of a given graph, while the latter can measure anything that the algorithm $\mathcal{A}$ produces. Their usage in our framework will be different and hence the distinction.

The null hypothesis H_0 throughout the paper is that for all graphs G that satisfy the given constraints, the values of $t(\mathcal{A}(G))$ follow the same null distribution. These constraints will be discussed in more detail later, when describing the statistic preserving randomization. Defining the null distribution is often the most challenging task in statistical hypothesis testing. This can be achieved by means of analytical expressions or by randomization tests.

The basic idea in randomization is to perturb the original data and carry out the experiments with the randomized version of the data. The randomization is controlled in such a way that the random datasets follow a certain distribution, which is typically chosen so that some properties (structural statistics) of the original data are maintained with a sufficient precision. When having several randomized datasets, the experiments on each one of them yield a set of values for the test statistic t, which follow the null distribution. These are used to define an *empirical p-value*: the fraction of test statistic values from the set that are at least as extreme than the test statistic value of the original data [24]. A significance test entails a definition of a significance level α, which is the maximum p-value allowed to reject the null hypothesis. We use $\alpha = 0.05$ throughout this paper.

4 The graph randomization problem

We would like to generate graph samples from a specified null distribution, and then, evaluate how the results obtained on the randomized graphs compare to the results on original (input) graph. An important feature of our framework is that the randomization method preserves certain structural properties (graph statistics) of the input graph during the perturbation process, and hence all random graphs will have the same structural properties. To justify this: suppose that the space of all graphs, up to some size, are equally likely and the randomization would thus return a completely new graph uniformly at random. This unrestricted null hypothesis would almost always be rejected. The reason for this is that practically all real world graphs have *some* structure that is lost in such a randomization. The data mining algorithm would thus find some structure in the original graph, and the result would be almost always significant when comparing it to the randomized versions of the input data.

A less naive solution is to sample only graphs that have the same number of nodes and edges with the original graph. This could be reasonable in some cases; however, still we would have the problem of not preserving many of the graph statistics that are important for describing the structure of the original graph. Hence, it is very likely that a data mining result would turn out to be significant also in such a setting. To sample graphs under additional constraints is reasonable: if the obtained results are not significant we can state that the constraints (let it be graph statistics or other conditions) explain the structure of the graph. On the other hand, the significance of the results would imply that the structures found by the data mining algorithm are not simply random artifacts that can be explained by the constraints.

Thus, the task we are addressing is the following.

PROBLEM 1. *Given an unweighted undirected graph G with n nodes and m edges and a set of graph statistics $\{r_1, \ldots, r_k\}$, generate a graph $\hat{G}$ chosen independently and uniformly from the set of graphs with n nodes and m edges and with the same degree distribution as G, which have approximately the same values for $\{r_1, \ldots, r_k\}$.*

We specify next what "approximately" means above.

4.1 Statistic preserving randomization

All the edge swaps to randomize the graph will naturally preserve exactly the number of nodes and edges, as well as the degree distribution. Before presenting the different types of local swaps and the associated algorithms, we discuss next how to restrict the randomization to preserve other graph statistics such as the characteristic path length or the average clustering coefficient. In many graph applications, maintaining the characteristic path length in randomization seems a natural choice

because, for example, of the six degrees of separation assumption in some social networks. On the other hand, the average clustering coefficient of a graph expresses the 'clusteredness' of the data, and will be appropriate in many applications. Still it is important to point out that the randomization algorithm we present is general enough, and it works with any combination of statistics on graphs. The proper choice will depend on the application.

A natural observation to justify our framework is the following: in many practical situations it is infeasible and unjustified to require that the graph statistics are maintained exactly. Devising a way to randomize graphs that preserves the exact value of a graph statistic, such as the average clustering coefficient or the characteristic path length, is virtually impossible. And even if it was possible to have a randomization algorithm that exactly preserves a value of a graph statistic, we would end up with a method difficult to generalize to other statistics. A further problem is that real world data is usually noisy and graph statistics should therefore be allowed some variance.

To allow for this approximate preservation while randomizing: let $\rho_0(G_s)$ be the distribution such that all graphs with a given number of nodes and edges and a certain degree distribution are equally likely. Our solution is to allow the user to define a distribution $\rho(G_s)$ from which the random samples will be drawn. In this paper, we will define $\rho(G_s)$ as a Gaussian distribution centered at the value of the preserved statistic of the original graph,

$$\rho(G_s) \propto \mathcal{N}(r(G_s) - r(G), 0, \sigma^2\mathbf{1})\rho_0(G_s), \tag{4.4}$$

where $\mathcal{N}(\cdot,\cdot)$ denotes a multidimensional Gaussian probability density function with a given mean vector and covariance matrix, and $r(G_s)$ and $r(G)$ describes a vector of values of the graph statistic in the sampled and original graphs, respectively. Note that if no graph statistic is specified, then at least the randomizations will be preserving the number of nodes and degree distribution of the original graph, as required by the definition of Problem 1. Also notice that r is overloaded here to represent a vector of values of the several statistics we wish to preserve. However, for the experiments later, we will choose to preserve only one statistic, in order to make more simple the discussion and understandability of the results.

In the following we will denote by Uniform the randomization that preserves *only* the degree distribution, that is, samples that come directly from ρ_0. The randomizations with additional constraints that preserve the average clustering coefficient and the characteristic path length are denoted by AvgCC and CPL, respectively. Notice that the distribution defined in Equation 4.4 is general enough to allow the use of other statistics apart from the ones studied here.

5 Algorithms

In this section we describe three MCMC sampling methods and show that they satisfy two important criteria, namely if the MCMC chain has converged, the samples are are drawn i.i.d. from $\rho(G_s)$; and even if the MCMC chain has not converged, the p-values we obtain are conservative.

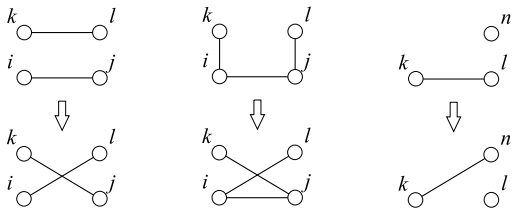
In the first part of the section we describe how to sample directly from from ρ_0, that is, obtaining sample graphs that preserve exactly the set of nodes and the original degree distribution only. This will be exactly the same as Uniform. In the second part we show how to further restrict the randomization to additionally preserve the other graph statistics.

5.1 Sampling from ρ_0: Three types of swaps

Our MCMC algorithms are based on performing one type of the three local edge swappings we will propose here. The basic idea is that we start the Markov chain from the original graph and make small random distortions that will affect at most two edges. These local distortions are designed to preserve the degree distribution of the original graph at all times. By means of applying these swaps as long as is needed for the chain to mix, we will arrive to a randomized graph, different from the original one.

The first swapping method follows the idea in [11], and has been previously used in [27]. The swap is to select two random edges and swap two endpoints together. The swap is dubbed XSwap for the shape of the swap, and it is illustrated in Figure 1a. The swap has the property that it maintains the individual node degrees as well as being very general. However, the swap does not maintain the connected components in the graph, and may therefore be not acceptable in some situations.

Consider a co-authorship graph where nodes are authors and edges signify that the authors have co-authored in a paper. When randomizing with the XSwap, a swap can be translated as an author collaborating with an author from, possibly, a completely different research area than he or she originally works in. This kind of mixing may not be desirable in some settings. To remedy that, we propose an another swap, called LocalSwap, that does not mix edges between connected components and respects the locality of connections. Figure 1b illustrates the LocalSwap. In the co-author example, a LocalSwap corresponds for an author k to change the co-authorship from an author i



(a) XSwap (b) LocalSwap (c) Flip

Figure 1: Different swaps for randomization. The Flip in c) is further conditioned with $|\delta(n) - \delta(l)| = 1$, where δ returns the degree of the node.

to an author j, with which i has worked. Hence, the swap respects the locality of the edges. However, observe that it can be shown that in a graph with a single connected component, each swap of XSwap or LocalSwap can be reproduced by a series of swaps of the other, excluding some special cases.

In some situations the individual node degrees do not matter and preserving them could be excessively restrictive. Therefore, we propose a third swap called Flip, which is illustrated in Figure 1c. A random edge and a node are selected. Then either endpoint is selected at random and the swap is done *if* the degree of the endpoint differs by one of the degree of the single selected node. This operation maintains the degree distribution, but changes the individual node degrees. Flip allows the graph to change more freely. As a special case, if the graph has a gap in the degree distribution, namely no nodes with a certain degree exist, but degrees of both smaller and larger do, then no edges are exchanged between the sets of nodes of small and large degrees.

Notice that the different swaps restrict the randomization in different degrees. The Flip is obviously the least restrictive since it does not maintain the individual node degrees, while the other swaps do. Among LocalSwap and XSwap, LocalSwap is more restrictive because it does not allow swaps between disconnected components — however, the difference is evident only if the original graph is disconnected.

Since the three swaps introduced above are small changes, we call two graphs *adjacent* if they can be reached from one another by a single swap. Using this definition of adjacency, each graph corresponds to a state in a Markov chain. The chain is reversible, in that for each single swap, we can perform a corresponding reverse swap. However, the chain might not be regular, i.e., it may be periodic. To guarantee regularity, we consider all illegal swaps to be self-loops to the current state in the Markov chain [11]. A swap is illegal, for instance, if it would result in duplicate edges. If we define the probability of a swap equally likely to the probability of its reverse swap, the steady state distribution of the regular Markov chain is uniform. Therefore, if the chain is randomized with any of the swaps, all the obtained graphs are equally likely.

5.2 Sampling from ρ: AvgCC and CPL If $\rho(G_s)$ is defined as Uniform, using one of these swaps with self-loops is all that is required for the Markov chain to converge to that distribution. However, if $\rho(G_s)$ is not Uniform, the swapping needs to be further controlled. As mentioned earlier, Metropolis-Hastings approach can be used to define state transition probabilities to make the Markov chain have the required steady state distribution. The swap from a graph G to a graph G' is performed with the Metropolis-Hastings probability $\min(1, \frac{\rho(G')}{\rho(G)})$, where $\rho(G)$ is defined by Equation (4.4). Because of this, the Markov chain will have the steady state distribution $\rho(G)$, which is the distribution we want to sample from.

Algorithm 1 outlines the randomization algorithm summarizing the three different types of swaps Flip, XSwap and LocalSwap, respectively. Notice that for compactness of the presentation, the outline of Algorithm 1 contains the three types of swaps. In a run of Algorithm 1 different types of swaps are never mixed.

Also note that if we assume that the user defined distribution for the graph statistics is strictly continuous, then all graphs with the same node degrees are reachable by the sampling algorithm. This is because for continuous distribution, all transition probabilities are greater than zero, and hence, given infinite time, all graphs are visited.

The computational complexity of one swap is $\mathcal{O}(\log |V|)$ if the node neighborhoods are represented by binary trees. The calculation of the graph statistics introduce much more complex work load. The average clustering coefficient requires a single calculation for all nodes and then the updates caused by swaps can be done locally. This requires a time of $\mathcal{O}(b)$, where b is the maximum degree of all nodes.

On the other hand, the calculation of the characteristic path length is $\mathcal{O}(|E||V|)$, if done from scratch every time. The calculation of the characteristic path length is a special case of calculating the all pairs shortest paths, which is a well known and much studied problem because of its fundamental nature and application, such as in routing [6]. For the experiments of this paper we adapt D'Esopo-Pape algorithm [1]. The adaption made for our problem is the following: a queue of nodes is initially filled with all the nodes in the graph and when

Algorithm 1 Randomization algorithm for graphs. Notice that for compactness of the presentation the algorithm contains the three types of swaps. In a run of the algorithm different types of swaps are never mixed.

Input: Undirected graph G, distribution ρ and number of steps T.
`Randomize`(G, ρ, T)
$G_s \leftarrow G$
repeat
  FLIP:
    Select an edge $(i,j) \in E(G_s)$
    Select (k,l) as either (i,j) or (j,i).
    Select node $n \in V(G_s)$
    if $k \neq n$ and $l \neq n$ and $(k,n) \notin E(G_s)$
      and $|\delta(k) - \delta(n)| = 1$ **then**
      $E(\widehat{G_s}) \leftarrow E(G_s) \backslash \{(k,l)\} \cup \{(k,n)\}$
      do with probability $\min(\rho(\widehat{G_s})/\rho(G_s), 1)$
        $G_s \leftarrow \widehat{G_s}$
      end do
  XSWAP:
    Select edges $(i,j),(k,l) \in E(G_s)$
    if $(i,l) \notin E(G_s)$ and $(k,j) \notin E(G_s)$ **then**
      $E(\widehat{G_s}) \leftarrow E(G_s) \backslash \{(i,j),(k,l)\} \cup \{(i,l),(k,j)\}$
      do with probability $\min(\rho(\widehat{G_s})/\rho(G_s), 1)$
        $G_s \leftarrow \widehat{G_s}$
      end do
  LOCALSWAP:
    Select edges $(i,j),(i,k),(j,l) \in E(G_s)$
    if $k \neq l$ and $k \neq j$ and $l \neq i$ and
      $(i,l) \notin E(G_s)$ and $(j,k) \notin E(G_s)$ **then**
      $E(\widehat{G_s}) \leftarrow E(G_s) \backslash \{(i,k),(j,l)\} \cup \{(i,l),(j,k)\}$
      do with probability $\min(\rho(\widehat{G_s})/\rho(G_s), 1)$
        $G_s \leftarrow \widehat{G_s}$
      end do
until reaching T steps
return G_s
Output: Randomized graph G_s

a node is popped out from the top of the queue, its neighbors are checked for violation of the condition

$$d_G(v,u) \leq d_G(w,u) + 1, \forall (v,w) \in E(G) \wedge \forall u \in V(G).$$

The condition states that the distance to a target node has to be at most the distance from your neighbor to that node plus one. If a neighbor violates this condition, its distance is updated and it is added to the bottom of the queue if it has not been in the queue after it was initially popped out. However if it has been in the queue at least two times, including the initial time, the node is inserted to the top of the queue. The heuristic idea is that if the node visits the queue for three or more times, it is likely that its neighbors need to be updated also. The algorithm does not have a guaranteed polynomial bound, but the exceptions are specially constructed cases. We chose to use this method because it was very simple to implement and faster than the alternatives.

5.3 Forward-backward sampling The problem in randomization with MCMC is that if the algorithm is merely run for all samples always using the original graph as the input graph, the samples are not considered to be independent since the they all are dependent on the input graph. We break the dependency by using backward-forward sampling [3, 4] by running the Markov chain backwards for T steps, and then run it forwards for T steps and use the last sample as a sample from the desired distribution. However, since the chain is reversible, this corresponds to first running the algorithm for T steps, use the resulting graph as the input graph for further randomizations and run the algorithm with that graph for T steps for the number of samples desired. We cannot guarantee that the MCMC has converged, but we can guarantee that we obtain conservative p-values (if the chain has not converged, the obtained p-values should be larger, see [2] page 46).

6 Graph mining applications

In the following sections we will present data mining applications in graphs and a proposed way of using our framework in defining the empirical p-values.

6.1 Clustering Graph clustering has received much attention [7, 26, 28, 34] (see also[1]). Most of the clustering methods define an objective function that has to be optimized to discover the final cluster structure in the data. Many methods focus on minimizing the number of edges between clusters while maximizing the edges within clusters. One way to solve this is to use spectral clustering [34], which is well known and often used method. It is not in the scope of this paper to evaluate all the different graph clustering algorithms. In the experimental section, we study spectral clustering of [34] and the kernel-based multilevel algorithm of [7].

In statistical testing, selecting the test statistic to measure is crucial to the validity of the test. Since all clustering methods optimize an essential objective function $f_{\mathcal{A}}(\mathcal{A}(G_0))$, where $\mathcal{A}$ is the clustering algorithm, we can use this value as a natural measure of the goodness of the results, that is $t(\mathcal{A}(G_0)) = f_{\mathcal{A}}(\mathcal{A}(G_0))$. The statistical testing consists of first calculating the value of the objective function $s_0 = f_{\mathcal{A}}(\mathcal{A}(G_0))$ for the result of the original graph, and then, computing the same for the set of graphs sampled from the desired distribution $s_i = f_{\mathcal{A}}(\mathcal{A}(G_i))$, where G_i is ith sampled graph. The empirical p-value is the fraction of samples s_i that are more extreme than s_0. If the p-value is at most the predefined threshold α the clustering on original graph is said to be statistically significant.

[1] http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/

	Nodes	Edges	AvgCC	CPL
Zachary	34	78	0.5706	2.3374
Adjnoun	112	425	0.1728	2.5129
Football	115	613	0.4032	2.4864
Power	4941	6594	0.0801	-

Table 1: Characteristics of the real datasets. AvgCC stands for average clustering coefficient, and CPL stands for characteristic path length.

6.2 Frequent graph pattern mining Another popular application for graphs is the mining for patterns in graphs. The algorithms have different objectives: some try to find patterns that appear often enough in a set of graphs [17], while others find patterns that compress the data well [5]. See [30] for a review of graph data mining methods. For our experiments, we focus on the algorithm presented in [17]; we use the implementation provided at the authors' web site. The algorithm finds frequently occurring subgraphs in a set of graphs. The frequency of occurrence is often called support, and it is similar to that in Apriori algorithm [14]. Again, for the computation of the p-value, we need to define an appropriate measure of goodness; we will use the support of each pattern for this purpose. Notice that in the case of pattern mining we will not evaluate the statistical significance of the individual patterns from our empirical p-value directly. Evaluating patterns is a problem of multiple hypothesis testing, which falls outside the scope of this paper. The scheme could be used to test the significance of the number of patterns, but not each pattern individually.

7 Experiments

We use five different real datasets[2]: Zachary [35], Adjnoun [22], Football [12], Power [31] and Compound[3]. Table 1 lists some characteristics of these datasets. Zachary is a social network of a karate club in US in 1970's. Adjnoun is an adjacency graph of common adjectives and nouns in a novel. Football contains a match graph of football teams in US colleges in 2000. Power dataset represents the power grid network of Western States in the US. And finally, Compound dataset contains 340 chemical compounds as graphs. The characteristics vary with different compounds; the largest graph has 214 nodes. CPL was not calculated for Power due to exceedingly long computation time.

We first performed a convergence analysis for few randomizations, and then randomized the whole dataset with the number of swaps found appropriate. The convergence was analyzed by running the algorithms a sufficient number of times and for each swap, the distance between the current perturbed graph G_s and the original G was calculated by

$$d_F(G_s, G) = \sum_{v \in V} (|\Gamma_G(v)| + |\Gamma_{G_s}(v)| - 2|\Gamma_G(v) \cap \Gamma_{G_s}(v)|).$$

Note that the set of nodes remains the same throughout the randomization. The distance corresponds to the square of the Frobenius norm of the adjacency matrices of the graphs. Figure 2 displays the convergence of all the algorithms with Adjnoun data and AvgCC statistic. The number of swaps was set to 500000, but since we run the algorithms backwards the equal number of steps, the total number of swaps is 1000000. Notice that the number of swaps correspond to the outer loop of the Algorithm 1, and thus it does not correspond to the number of actual changes made to the graph because of the self-loops and probabilities for swapping. The Markov chain is considered to have converged when the distance has stabilized to around some value. The convergence results correspond to the expectations: both LocalSwap and XSwap converged similarly and faster than Flip. The number of self-loops, illegal swaps, in Flip is usually much greater than that of LocalSwap and XSwap. Therefore, Flip is much slower to converge. However, the computationally costly portion of the algorithms is updating the graph statistics. The speed of convergence therefore is not the determining factor for the running time of the algorithm. Notice also that Flip gains more distance to the original graph than the other algorithms since it is not restricted to maintain individual node degrees, and therefore, is able to operate more freely.

For all randomizations throughout the experiments, the variance for the graph statistic was set to $\sigma^2 = 10^{-7}$. We found this value to allow the randomization to vary enough for the randomized graphs to gain enough distance from the original, while maintaining the value of the statistic. Figure 3 displays the histogram of the value of the CPL statistic for 100 randomized graphs with Football data. The variation is well within the allowed variance.

We also measured the execution times of our algorithms to produce a single random graph. Table 2 lists these times for the Adjnoun dataset. Each algorithm with each statistic was run long enough for it to converge. The number of swaps performed are listed in Table 3. The times are not strictly comparable, but they give a rough idea of how long it takes to obtain a random graph.

[2] http://www-personal.umich.edu/~mejn/netdata/

[3] http://www.doc.ic.ac.uk/~shm/Software/Datasets/carcinogenesis/progol/carcinogenesis.tar.Z

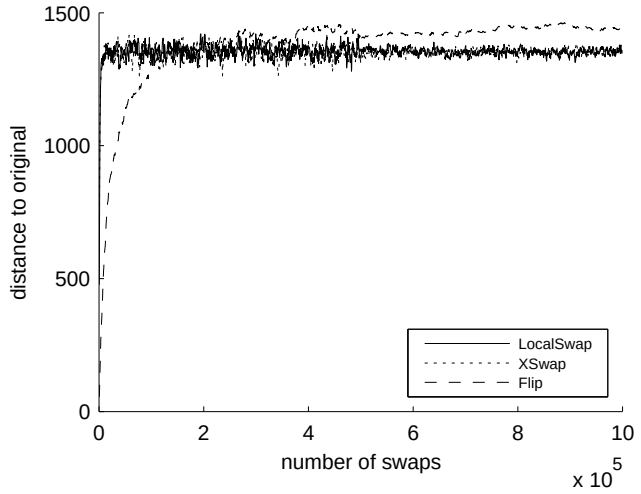


Figure 2: Convergence of the algorithms with Adjnoun data and AvgCC statistic. The change in the middle is because the first half is a single graph, the reverse direction, while after the halfway, five graphs were randomized and the figure depicts their mean distance from the original graph. The variance was very small for the distances of the five graphs.

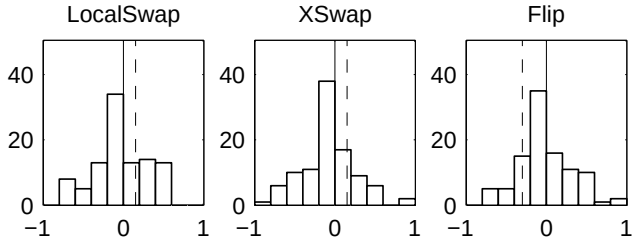


Figure 3: Histogram of CPL statistic for different algorithms with Football data and 100 samples. The horizontal axis represents the difference from the original statistic value multiplied by 10^4. The variance of the Gaussian distribution, centered at the original value of the statistic, was set to 10^{-7} (standard deviation being roughly $3 \cdot 10^{-4}$). The continuous vertical line is set to 0, expressing the original value for the statistic, while the dashed line represents the value of the statistic for the graph after reverse swaps.

7.1 Graph clustering Using the Zachary, Adjnoun, Football and Power datasets, 100 random graphs were generated for all pairs of algorithms and statistic; except for Power and CPL due to the dataset being too large to use with CPL in a reasonable time. Before every randomization, a convergence analysis was performed to find the number of swaps needed for convergence.

	Uniform	AvgCC	CPL
LocalSwap	9.6ms	110ms	24s
XSwap	13ms	122ms	31s
Flip	45ms	140ms	8s

Table 2: Computing times of a single random graph in seconds of Adjnoun data with different statistics. Notice that these times also contain stepping to the reverse direction, which is roughly half of the time. This time would not accumulate when taking several random graphs.

Note that because the algorithm, dataset and statistic to preserve varies, the convergence analysis needs to be carried out separately for each combination. Table 3 lists the number of swaps used for randomization.

After 100 random graphs were generated, the different samples were clustered with Graclus [7], readily available from the authors' site[4], as well as a spectral graph clustering method [34]. Both methods minimize the cut between clusters, and hence, they should produce similar results. The algorithms were used to cluster individual graphs from 2 to 15, 30, and 50 clusters, for Zachary, Adjnoun and Football, and Power datasets, respectively. For each clustering, the minimum cut value, the value of the objective function, was stored. Finally, the empirical p-value for each combination of algorithm, dataset and statistic was calculated as described above. The results for the different datasets follow next.

Zachary The empirical p-values were similar between the combinations of different graph statistics and randomization algorithms. With Uniform distribution, the p-values for small cluster numbers were slightly smaller than with either AvgCC or CPL. However, whether or not the p-values were less than the $\alpha = 0.05$ threshold remained more or less unchanged. With all randomization algorithms and graph statistics, the p-value for two clusters was below the α threshold, as it should be, since we know there are two clusters in the data. We also discovered two interesting phenomena, seen in Figure 4. The p-value rapidly increases to around one at 8 clusters and stays there for several number of clusters. The reason for this is that when the original data is clustered, there is a limit to how many reasonable number of clusters the graph can be divided to. When this value is exceeded, the algorithm has to produce cluster borders within clusters, which results in a high value for the objective function. Since the random graphs do not have this structure, adding one cluster more makes no big difference. Hence the rapid incline.

[4] http://www.cs.utexas.edu/users/dml/Software/graclus.html

	LocalSwap			XSwap			Flip		
	Uniform	AvgCC	CPL	Uniform	AvgCC	CPL	Uniform	AvgCC	CPL
Zachary	2	300	200	2	100	100	5	300	500
Adjnoun	10	50	100	10	50	100	100	300	500
Football	5	4000	200	5	4000	200	20	5000	200
Power	200	5000	-	400	5000	-	2000	5000	-

Table 3: Number of swaps used for different combinations of dataset, algorithm and statistic. The values have been multiplied by 10^{-3} for clarity.

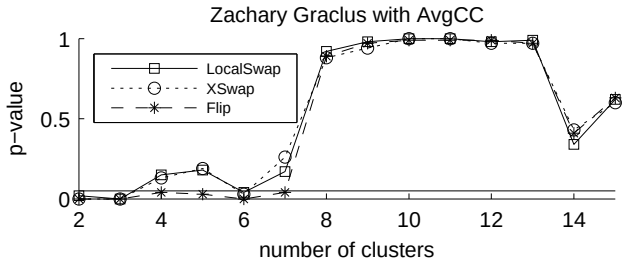


Figure 4: Empirical p-values of the Graclus results wrt number of clusters with Zachary dataset, all randomization algorithms and AvgCC as graph statistic. The continuous horizontal line represents the $\alpha = 0.05$ confidence threshold.

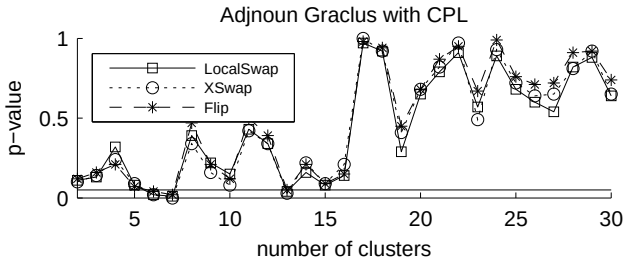


Figure 5: Empirical p-values of the Graclus results wrt the number of clusters with Adjnoun dataset, all randomization algorithms and CPL as statistic. The continuous horizontal line represents the $\alpha = 0.05$ confidence threshold.

The other phenomenon is that p-values for spectral clustering did not exceed the α threshold around 5 clusters. Graclus seems to have trouble clustering to around five clusters, which could be caused by the approximate nature of the algorithm in combination with this dataset.

Adjnoun The Adjnoun dataset is not known to contain clusters, and by visual inspection of it, there are none. The graph contains few highly connected nodes with very large degrees, while other nodes have only few links. The highly connected nodes seem to be all connected between themselves and other high degree nodes, so no clustering structure is evident among them. Again, there were no large differences between the combinations of randomization algorithms, graph statistics and clustering algorithms. However, as illustrated in Figure 5, the p-values vary greatly when the number of clusters is changed. This is consistent with the assumption that the clustering finds no statistically significant structure from the Adjnoun.

Football The empirical p-values for the Football data were practically the same across the Graclus and the spectral graph clustering algorithms. However, this dataset exhibited also an interesting phenomenon, which can be seen in Figure 6. The two randomization algorithms, LocalSwap and XSwap, behave differently from Flip when the number of clusters is small and the randomization is constrained with AvgCC. For CPL and Uniform, the results are almost identical among the three randomization methods. The reason for this to happen is that Flip is less constrained than LocalSwap and XSwap, and the difference becomes explicit with this dataset. Flip only preserves the degree distribution, while the other algorithms also preserve the individual node degrees. And since the degrees of nodes in this dataset have only a few values, all between 7 and 12, Flip has a lot of freedom to make changes to the graph.

Power The results were always below the α threshold for all number of clusters from 2 to 50, with no variation among randomization and clustering algorithms, even with added constraint for AvgCC. In all, the dataset has a clear structure that the clustering algorithms found.

7.2 Pattern Mining We first ran experiments with an artificially created dataset having 10 given graph patterns with labels on the nodes. We created a dataset of 100 graphs by randomly adding edges and nodes, with random labels, around a randomly selected subset of the 10 original patterns. Our intuition is that the subgraph mining algorithm should find the original 10 patterns.

We ran the convergence analysis for the dataset for both Uniform and AvgCC. The number of swaps de-

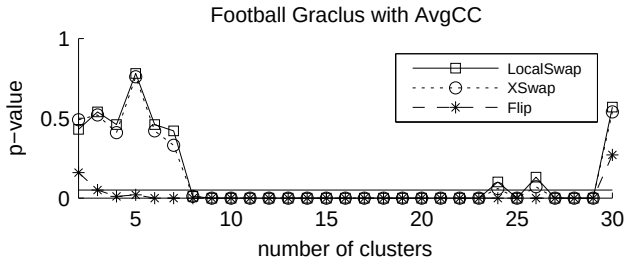


Figure 6: Empirical p-values of the Graclus results wrt the number of clusters with Football dataset, all randomization algorithms and AvgCC as statistic. The continuous horizontal line represents the $\alpha = 0.05$ confidence threshold.

	Artificial		Compound	
	Uniform	AvgCC	Uniform	AvgCC
LocalSwap	3	10	50	50
XSwap	1	10	10	10
Flip	10	10	100	100

Table 4: Number of swaps for the artificial and compound datasets multiplied by 10^{-3}.

termined by the converge analysis are listed in Table 4. Then, 100 randomizations of the sets of graphs were generated by randomizing each graph individually using all the randomization algorithms.

We used the FSG [17] algorithm, which is a part of Pafi[5], to find the frequent subgraphs in this database. We run the tests with different minimum support values to see if there is any difference. From the results with the original graphs, frequent patterns were stored as well as their support. The empirical p-value of the pattern was computed for each pattern, using the respective support as a test statistic.

We performed the same experiments for the Compound dataset.[6] The fraction of patterns under the α threshold are shown in Figure 7. Note that the additional constraint of restricting the randomizations to maintain AvgCC does not have much effect in the results. The results with the artificial dataset were similar.

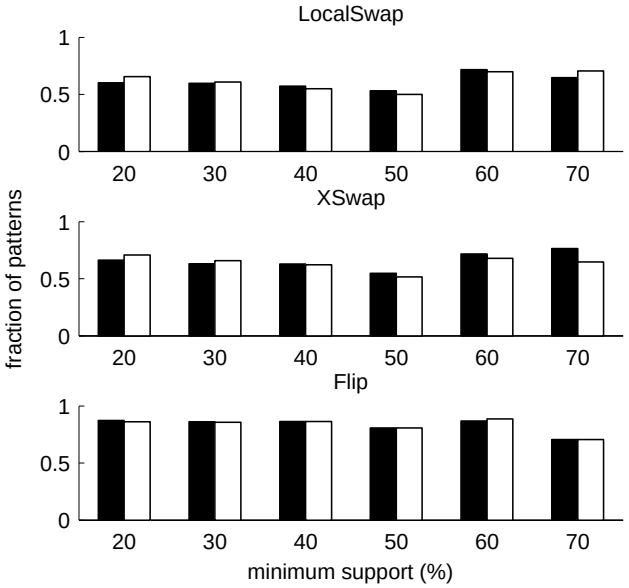


Figure 7: Graph mining results for all the algorithms with Compound data. The horizontal axis lists the minimum supports used. The height of a bar represents the ratio of patters whose p-value is under the α threshold 0.05. Black bars correspond to the results with Uniform, while white bars correspond to the results with AvgCC.

8 Discussion and conclusions

Within the framework of statistical significant testing, randomization techniques allow for a generation of a set of random data samples drawn from a specified null distribution about the input data. In this paper we focus our efforts on randomization techniques for unweighted undirected graphs. We propose three edge swapping methods that, together with Metropolis sampling techniques on the input graph, result in three algorithms for sampling randomly graphs from a user-specified null distribution. Our framework is flexible enough because the null distribution can be set according to the problem: random samples are always drawn from the set of graphs with the same degree distribution as the original graph, and additionally, the user can specify other statistics to preserve. Here we showed how to preserve average clustering coefficient and characteristic path length.

Once the samples from the null distribution are generated, calculating the empirical p-value follows the standard definition, i.e., the fraction of test statistic values from the set of random samples that are more extreme than the test statistic value of the input data. For graph clustering, the test statistic used here is the value of the objective function; for frequent subgraph mining, the test statistic is associated individually to each pattern and it can be, e.g., its frequency in the data. Evaluating significance in clustering is direct and it corresponds to checking whether the calculated p-value is

[5] http://glaros.dtc.umn.edu/gkhome/pafi/overview

[6] Notice that the randomizations may violate some physical laws, such as the valency numbers of atoms in a molecule may not be preserved. Despite this, we use the dataset as an example of how to use the randomization method for real data.

under a given α significance level. However, evaluating significance for individual patterns in frequent graph discovery is not straightforward as it falls in a multiple hypothesis problem [8, 32]. This problem is outside the scope of this paper and we do not discuss it further; still, the randomization framework proposed here is general and flexible enough to allow for future studies on the multiple hypothesis testing problem.

The utility of the framework, as well as how the p-value vary in several datasets, is shown in the experimental section. We saw that the choice of the null hypothesis can change the testing results in some cases. This is due to having a more restrictive null hypothesis, which is coupled with the graph statistics that samples share with the original data. We have not discussed so far which is the choice of the null hypothesis for a given application. Obviously, the more restricted null hypothesis – more statistics are specified to be preserved in the samples – the less significant the results of a data mining algorithm tend to be. In this sense, the randomizations proposed here are useful towards evaluating the following question: do the observed data mining results on the original graph convey any information in addition to the specified statistics?

From this perspective the null hypothesis depends on the user. For instance, in the clustering application it is important to know whether the discovered clusters provide some structural information different from the average clustering coefficient of the graph or the characteristic path length. Obviously, for an input graph where groups of nodes are all very distant from one another, clusters can be easily discovered by any algorithm. However, characteristic path length alone could give an idea of the structure in the network and thus, the results of a clustering algorithm might not be significant enough under those conditions. The same example can be given for pattern discovery on graphs. A more sensible null hypothesis there is to know whether degree distribution already tells something about the subgraph patterns found in the data. For example, patterns consisting of one single node would not be significant then.

Finally, it is worth noting that it would be fairly easy to extend the techniques presented here to directed graphs; yet, handling weighted graphs and considering edge or node labels is not straightforward. This will be an interesting future direction. More importantly for future work, we will explore how the combinations of the statistics restrict randomization and how conclusions can be derived from those more restricted samples.

References

[1] Dimitri P. Bertsekas. *Network Optimization: Continuous and Discrete Model*. MIT Press, 1998.

[2] Julian Besag. Markov chain Monte Carlo methods for statistical inference. http://www.ims.nus.edu.sg/Programs/mcmc/files/besag_tl.pdf, 2004.

[3] Julian Besag and Peter Clifford. Generalized Monte Carlo significance tests. *Biometrica*, 76(4), 1989.

[4] Julian Besag and Peter Clifford. Sequential Monte Carlo p-values. *Biometrica*, 78(2), 1991.

[5] J. Cook and L. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255, 1994.

[6] Camil Demetrescu and Giuseppe F. Italiano. A new approach to dynamic all pairs shortest paths. *Journal of the ACM*, 51(6), 2004.

[7] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. A fast kernel-based multilevel algorithm for graph clustering. In *Proc. of the 11th ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2005.

[8] Sandrine Dudoit, Juliet Popper Shaffer, and Jennifer C. Boldrick. Multiple hypothesis testing in microarray experiments. *Statistical Science*, 18(1), 2003.

[9] Eugene S. Edgington. *Randomization Tests*. Marcel Dekker, Inc., New York, 3rd edition, 1995.

[10] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2), 2005.

[11] Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas. Assessing data mining results via swap randomization. In *Proc. of the 12th ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

[12] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12), 2002.

[13] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15, 2007.

[14] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. MIT Press, 2001.

[15] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrica*, 57, 1970.

[16] Mehmet Koyutürk, Wojciech Szpankowski, and Ananth Grama. Assessing significance of connectivity and conservation in protein interaction networks. *Journal of Computational Biology*, 14(6), 2007.

[17] Michihiro Kuramochi and George Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. Knowl. Data Eng.*, 16(9), 2004.

[18] Pierre Legendre and Louis Legendre. *Numerical Ecology*. Elsevier Science, 1998.

[19] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *SICKDD International Conference of Knowledge Discovery and Data Mining*, 2006.
[20] Bryan F.J. Manly. *Randomization, Bootstrap And Monte Carlo Methods in Biology.* Chapman & Hall, 3rd edition, 2007.
[21] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21(1), 1953.
[22] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74, 2006.
[23] M.E.J. Newman. The structure and function of complex networks. *SIAM Review*, 45, 2003.
[24] B. V. North, D. Curtis, and P. C. Sham. A note on the calculation of empirical P values from Monte Carlo procedures. *The American Journal of Human Genetics*, 71(2), 2002.
[25] Markus Ojala, Niko Vuokko, Aleksi Kallio, Niina Haiminen, and Heikki Mannila. Randomization of real-valued matrices for assessing the significance of data mining results. In *Proc. of the 2008 SIAM International Conference on Data Mining*, 2008.
[26] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1), 2007.
[27] Roded Sharan, Trey Ideker, Brian Kelley, Ron Shamir, and Richard M. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *Journal of Computational Biology*, 12(6), 2005.
[28] A.J. Soper, C. Walshaw, and M. Cross. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. *Journal of Global Optimization*, 29, 2004.
[29] Yang Wang, Chakrabarti, Chenxi Wang, and Christos Faloutsos. Epidemic spreading in real networks: an eigenvalue viewpoint. In *Proc. 22nd International Symposium on Reliable Distributed Systems*, 2003.
[30] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *ACM SICKDD Explorations Newsletter*, 5(1), 2003.
[31] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world. *Nature*, 393, 1998.
[32] Peter H. Westfall and S. Stanley Young. *Resampling-based multiple testing: examples and methods for p-value adjustment.* Wiley, 1993.
[33] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *Proc. of the 2008 SIAM International Conference on Data Mining*, 2008.
[34] Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. In *Proc. of the Ninth IEEE International Conference on Computer Vision (ICCV'03)*, 2003.
[35] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33, 1977.