Different Scales of Randomness: Empirical Mixing Times of the Edge Switching and Curveball MCMC

Deepak Ajwani
University College Dublin, Ireland

Melvin Kallmayer
Goethe University Frankfurt, Germany

Alexander Leonhardt
Goethe University Frankfurt, Germany

Ulrich Meyer
Goethe University Frankfurt, Germany

Ryan O' Connor
University College Dublin, Ireland

Manuel Penschuck
University of Southern Denmark, Odense, Denmark

Abstract

The Fixed Degree Sequence Model (FDSM) asks for a uniform sample from the set of all simple graphs that match a prescribed degree sequence. It is typically implemented using Markov-Chain Monte-Carlo (MCMC) processes, such as Edge Switching or Curveball (and their variants). Yet despite decades of research, rigorous bounds on the mixing times of such processes remain impractical.

Consequently, several experimental techniques have been used to derive "empirical lower bounds" on the mixing time. We address the following research questions: (1) Which commonly studied graph-theoretic properties serve as reliable empirical predictors for mixing of FDSM MCMC processes? (2) At what structural scales do these properties operate primarily (i. e., are they predominantly local or global in nature)? (3) How can these properties be characterised and quantified most effectively?

To this end, we propose Claim, a novel systematic method to establish empirical lower bounds using learnt classifiers, and compare it to existing methods. Apart from interesting insights into the usage of machine learning for this problem, we also derive robust graph properties with respect to different randomisation algorithms. Although experimental in nature, these results may influence both theorist's and algorithm engineer's work on improved bounds and better algorithm respectively.

2012 ACM Subject Classification Theory of computation → Random network models

Keywords and phrases Mixing Time, Graph Randomization, Machine Learning, Edge Switching

Digital Object Identifier 10.4230/LIPIcs.SEA.2026.2

Supplementary Material *Software*: `https://zenodo.org/records/19508271`

Funding The research of the first and penultimate author is supported in part by a grant from Science Foundation Ireland (Grant number 18/CRT/6183). For the purpose of Open Access, the authors have applied a CC BY public copyright license to any author-accepted manuscript version arising from this submission. This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG) – ME 2088/5-2, FOR 2975; and the Novo Nordisk Foundation grant NNF21OC0066551. The authors gratefully acknowledge the computing time provided to them at the NHR Center NHR@SW at Goethe University Frankfurt. This is funded by the Federal Ministry of Education and Research, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities (www.nhr-verein.de/unsere-partner).

Acknowledgements We thank the anonymous reviewers for their valuable and insightful feedback.

1 Introduction

Obtaining a *uniform* random sample from the set of all *simple* graphs adhering to a prescribed degree sequence $\mathcal{D}$ is a long-standing partially open problem both in theory and practice. The task is crucial in the construction of null models (e. g. [52, 34]) or as a routine in graph generators, e. g. the popular LFR benchmark [46, 45]. Let us define the problem formally:

▶ **Definition 1.** *Given a sequence $\mathcal{D} = (d_1, \ldots, d_n)$, let $\mathbb{G}(\mathcal{D})$ be the set of all simple graphs (no loops or multi-edges) where node v_i has degree d_i. We call $\mathcal{D}$* graphical *iff $\mathbb{G}(\mathcal{D}) \neq \emptyset$.*

▶ **Definition 2.** *Let $\mathcal{D}$ be a graphical degree sequence. We denote the uniform distribution of $\mathbb{G}(\mathcal{D})$ as $\mathcal{G}(\mathcal{D})$. The* Fixed Degree-Sequence Model (FDSM) *asks to sample from $\mathcal{G}(\mathcal{D})$.*

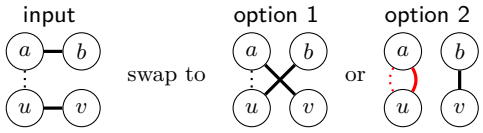
It is also common to start with some explicit graph $G_0 \in \mathbb{G}(\mathcal{D})$, thereby implicitly fixing $\mathcal{D}$. As discussed in Section 2, there exist simple linear-time algorithms to translate a graphical degree sequence into a non-random graph and vice versa. We therefore treat both problems synonymously and call them FDSM. The problem and its variants are also often discussed in terms of binary matrices with prescribed column and row sums (e. g. [66, 10]).

Although there are exact $\mathcal{G}(\mathcal{D})$ sampling algorithms for certain classes of degree sequences $\mathcal{D}$, the FDSM is often implemented using Markov-Chain Monte-Carlo (MCMC) methods. These methods typically inflict a large number of local changes to the graph. The Edge Switching Markov Chain, for instance, selects two edges uniformly at random and exchanges one of their endpoints each (see Figure 1a). For an appropriate MCMC, we converge towards the uniform distribution $\mathcal{G}(\mathcal{D})$. The number of steps τ required until we can treat G_i with $i \geq \tau$ as a sample from $\mathcal{G}(\mathcal{D})$, is called *mixing time*. Conceptually, τ is the number of steps the MCMC takes to "forget" its starting point.

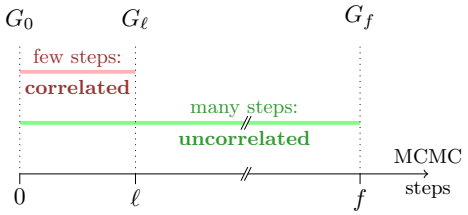
We say that a MCMC process is *rapidly mixing* if τ scales polynomially in the graph size. Even if the considered processes are known to be rapidly mixing for large classes of degree sequences (e. g. [23]), the existing rigorous bounds almost always involve impractically high-degree polynomials in both the graph size and the maximum degree. In practice, users often pick between $10m$ and $100m$ (e. g. [26, 25, 18]). In contrast, several empirical studies (e. g. [55, 53, 59, 2, 18, 32]) aim to determine the number of steps required until no "additional" randomisation is observed. These methods only yield a minimum number of required MCMC steps. Hence, we refer to these types of results as *empirical lower bounds*. Accordingly, and in line with the theoretical notion of mixing time, these *empirical lower bounds* denote the minimum number of steps required such that, with respect to a chosen set of features, no further randomization is detectable, aside from a negligibly small error.

Virtually all empirical approaches that are applicable to graphs of non-trivial size focus on particular structural properties of the graph in order to obtain their estimates. This naturally leads to the question of which properties are most appropriate for this purpose. A closely related issue concerns the scale at which such properties should be evaluated.

Roughly speaking, local properties produce fine-grained insights, but are susceptible to noise and may fail to capture global structure, whereas global properties tend to smooth out or obscure local phenomena. For instance, the effective autocorrelation method (see Section 3) analyses edges in isolation. This yields a valid empirical lower bound, since, in this context, local randomness is clearly a necessary condition for global randomness. On the other hand, it is well established that seemingly random local behaviour does not, in general, guarantee global randomness. This intuition is formalised through a number of mathematical concepts, such as low-order marginals, local patterns, and k-wise independence.



(a) Edge Switching selects two edges at random and switches their endpoints. A switch is rejected, if it introduces a non-simple structure (option 2).



(b) Hypothesis: (G_0, G_ℓ) and (G_0, G_f) are indistinguishable. If a classifier distinguishes both (rejects the hypothesis), ℓ is too small to properly mix.

Figure 1 Rewiring of the Edge Switching MCMC (left) and CLAIM's hypothesis (right).

1.1 Our contributions

We investigate (i) which established graph properties are good proxies to quantify the mixing of FDSM MCMC processes and (ii) whether they are local or global properties. To this end, we use multiple techniques, operating on different scales, to establish empirical lower bounds.

- Autocorrelation (Section 3) is an established and robust method that serves as a baseline. We systematically investigate its parameters and a novel generalisation.
- We propose the novel ML-based method CLAIM (Section 4) to systematically evaluate the importance of different graph properties. As illustrated in Figure 1b, the core idea is to test and reject the hypothesis that a given number of MCMC steps ℓ is sufficient to uniformly randomise the graph: Given a start graph G_0, we run ℓ steps of the process to obtain the graph G_ℓ. Analogously, we obtain G_f with $f \ggg \ell$.
  Our hypothesis is that (G_0, G_ℓ) remain *correlated*, while (G_0, G_f) are *uncorrelated*. We produce a large number of appropriately labelled graph pairs to train a classifier. If the classifiers distinguish both classes successfully, it indicates that (G_0, G_ℓ) share similarities that the classifier can recognise. This provides strong evidence that ℓ steps are *not* enough to obtain a sample from $\mathcal{G}(\mathcal{D})$ – this establishes an empirical lower bound on $\hat{\tau}$.
- Finally, we conduct statistical tests (Section 5) on the data.

We find that the methods yield consistent bounds despite their different foundations. This adds credence to the reliability and robustness of the results. Furthermore, we demonstrate practically significant performance differences between Edge Switching and Curveball, and provide evidence suggesting that common practical recommendations should be rephrased.

2 Related work

The Fixed Degree-Sequence Model (FDSM, see Definition 1) implies three requirements. The output has to be a (i) *simple* graph, (ii) match $\mathcal{D}$, and (iii) be a uniform sample. It is relatively straightforward to achieve any two of these properties. For completeness, we start by surveying relaxations and alternative sampling approaches.

2.1 Alternative sampling techniques

The Configuration Model [9, 13] can be implemented in linear time, but may emit multi-edges and self-loops. Another common choice is the Chung-Lu model [20, 21] which can produce simple graphs, however, its output matches $\mathcal{D}$ only in expectation.

The sampling of graphs that are simple and exactly match $\mathcal{D}$ is notably more challenging. An early approach repeatedly samples from the Configuration Model rejecting non-simple graphs [68]. Although it has a runtime linear in the number of edges, on regular graphs the runtime scales exponentially in the square of the maximum degree d_{max}; i. e., it is only polynomial time for $d_{max} = o(\sqrt{\log n})$.

The rejection rate can be significantly reduced by "repairing" non-simple graphs produced by the Configuration Model. [50, 51] Here, one identifies a sufficiently large class of graphs produced by the Configuration Model (e. g. few self-loops and multi-edges) and then attempts to remove illegal structures by swapping the end points of selected edges. It is crucial to perform these swaps in a manner that avoids introducing biases, often achieved through carefully balanced rejection steps. Notable examples include sampling of graphs with degrees up to the fourth root of the number of edges [51, 7], regular graphs with slightly higher degrees [30, 7], and power-law graphs with a sufficiently small power-law exponent [31, 7].

There are severe practical limitations to such approaches. Most crucially, they are highly non-trivial to design and implement – for instance, Allendorf et al. [3] discuss a practical implementation for power-law degree sequences with exponent of $\gamma \gtrapprox 2.88$. The code consists of more than 4000 lines of code and involves non-trivial arbitrary precision arithmetic. While it performs exceptionally well on instances matching the narrow parameter range, even slight deviations from the specification result in prohibitively high rejection rates.

2.2 Markov-Chain Monte-Carlo methods

The FDSM is tightly related with the degree-preserving randomisation of a graph, as translating between an instance $G \in \mathbb{G}(\mathcal{D})$ and $\mathcal{D}$ is possible in linear time in both directions. Given a graph, we can compute $\mathcal{D}$ by counting the degrees. Conversely, we can construct *some* simple graph $G \in \mathcal{D}$ using the deterministic Havel-Hakimi algorithm [39, 37].

Once a graph is obtained or provided, it can be randomised using Markov-Chain Monte-Carlo (MCMC) methods (e. g. [41, 22, 32, 35, 42, 48, 62, 63, 64, 65] where some chains introduce additional constraints such as connectivity or joint-degree distributions). These schemes carry out large number of steps, each inflicting only small local changes. In this article, we consider the following three common processes.

1. *Edge-Switching* [58] selects two edges uniformly at random with replacement and exchanges their endpoints as in Figure 1a. Switches creating loops or multi-edges are rejected without replacement. A rejection as such represents a self-loop in the state-graph of the underlying MCMC process.
2. *Curveball* [63] uniformly selects two nodes u and v and randomly exchanges some of their disjoint neighbours; this is called a *(Curveball) trade*. More specifically, let $N_{-v}(u) = N(u) \setminus N(v)$ and $N_{-u}(v) = N(v) \setminus N(u)$ be the disjoint neighbourhoods of both nodes. Then, a trade randomly redistributes the neighbours $X = N_{-v}(u) \cup N_{-u}(v)$ without changing the degrees of u and v. This is typically implemented by drawing $|N_{-v}(u)|$ elements from X without replacement and assigning them u, leaving the remainder for v.
3. *Global Curveball Markov-Chain* [17, 18] extends Curveball by grouping multiple single trades into a so-called *global trade*. A global trade is a sequence of up-to $n/2$ trades, s.t. each node is traded at most once. This latter constraint allows efficient parallelisation [18].

It is straightforward to show that these processes eventually converge to $\mathbb{G}(\mathcal{D})$ by establishing that the underlying Markov chains are ergodic and symmetric (see [16] for a detailed survey). Yet, rigorous and practical bounds on the mixing time (i. e., the number of steps until the process is sufficiently close to a uniform distribution) are mostly elusive.

While polynomial upper bounds [22, 24, 36, 23, 5, 29, 23] on the mixing time are known for several classes of undirected graphs such as bounded-degree or power-law graphs, the involved polynomials tend to be of impractically high degree. This is problematic, as the empirical indicators used to detect insufficient mixing fade rapidly, giving rise to a large gap between theory and practice; ultimately lead to the previously mentioned recommendation of employing an asymptotically linear number of steps to ensure adequate mixing.

One may ask whether a simple argument already yields a superlinear lower bound for the mixing time of the Curveball process. For instance, consider a graph with constant average degree but a small number of high-degree vertices with degree of order $\Theta(\sqrt{n})$, as in power-law graphs with exponent $\gamma = 3$. Such a vertex is selected with probability $\Theta(1/n)$, and in each step only a constant number of its neighbours are changed in expectation. This suggests a mixing time of $\Omega(n^{3/2})$. However, this reasoning applies only to the directed version of Curveball. In the undirected case, which we consider here, trades involving neighbouring vertices can also affect the high-degree vertex itself, thus the probability that the neighbours of a high-degree vertex changes is proportional to its degree, invalidating such arguments in this setting.

2.3 Empirical methods quantifying mixing

Rechner et al. [60] empirically study the full state graph of several MCMC processes and find that the spectral gap of the MCMC state graph tends to be a good proxy to the mixing time. Unfortunately, due to the combinatorial explosion of the state graph, this approach is applicable only for very small graphs. Various empirical approaches have been proposed to quantify the mixing time of MCMC approaches by tracing the random walks carried out. To our knowledge, data-driven methods based on autocorrelation time are the most sensitive; we discuss and generalise this method in Section 3.

An important class of methods consider the (approximate) distribution of graph properties, such as assortativity coefficients, clustering coefficients, distance measures, eigenvalues, triangle count, other motifs, perturbation and number of successful swaps [59, 38, 66, 16, 15]. The distribution of these properties is then interpreted as a proxy for the distribution of graphs. This "projection" is by construction lossy, and hence may overestimate the mixing achieved [10] – thereby underestimating the mixing time.

3 Autocorrelation

Pinar et al. [59, IV.A.] propose a data-driven approach based on autocorrelation time [61]. It is often more sensitive than the methods discussed in Section 2.3. Consider a long run of an MCMC process with a large number of steps N. For each step $1 \leq t \leq N$ and pair of nodes $\{u, v\}$, we define $z^{(t)}_{\{u,v\}} = 1$ if the nodes are adjacent at step t and 0 otherwise. This results in $\binom{n}{2}$ binary sequences $Z_{\{u,v\}} = (z^{(1)}_{\{u,v\}}, \ldots, z^{(N)}_{\{u,v\}})$.

Since most MCMC processes inflict only small changes in each step t, the values $z^{(t)}_{\{u,v\}}$ are *autocorrelated* (i. e., correlated to their predecessors $z^{(t-1)}_{\{u,v\}}$). For instance, Edge Switching affects at most four edges per step (two removals and two insertions). Hence, for any step t, there are at most four distinct sets $S_i = \{u, v\}$ for some vertices u, v for which $z^{(t)}_{S_i} \neq z^{(t+1)}_{S_i}$ holds, while the remaining, at least $\binom{n}{2} - 4$ pairs, match.

Pinar et al. consider a *k-thinned* subsequence $Z^{(k)}_{\{u,v\}} = (z^{(k)}_{\{u,v\}}, z^{(2k)}_{\{u,v\}}, \ldots)$ that is obtained from $Z_{\{u,v\}}$ by only considering every k-th entry. Intuitively, the autocorrelation of *thinned* sequences $Z^{(k)}$ vanishes for sufficiently large spacings k, since "k times" more changes are done. Different variants of this approach have been used [62, 59, 18, 2]. We adopt

the Bayesian Information Criterion (BIC) previously used by Pinar et al. [59] to classify k-thinned sequences as either *correlated* (i. e., best explained by a first-order Markov model) or *uncorrelated* (i. e., sequence entries seem independent). Thus, we classify each k-thinned sequence by evaluating $\Delta\text{BIC} = \text{BIC}^I - \text{BIC}^M$, where the superscript I and M indicates the independent and first-order Markov model, respectively. The sign of ΔBIC then determines the assigned class for each sequence.

3.1 Implementation details

We adopt the high-level structure of the original implementation[1] of Pinar et al. [59]. It consists of two phases. The first phase executes the MCMC process and records the binary sequences in time and memory $\Theta(\binom{n}{2}N)$. In the second phase, the authors consider each sequence individually. They apply the BIC to classify each of its K thinned subsequences as correlated or uncorrelated. This takes time $\mathcal{O}(\binom{n}{2}NK)$ where K is the number of thinning values considered (typically a small constant $K \lll N$). We improve the pipeline's complexity to $\mathcal{O}(K\binom{n}{2} + NKm)$ time and $\mathcal{O}(K\binom{n}{2})$ memory by modifying the data model:

- The second phase requires only four values per k-thinned sequence: x_{00}, x_{01}, x_{10}, and x_{11} where x_{ij} encodes the number of transitions from i to j in adjacent bits. Instead of explicitly recording large sequences, we maintain $4K$ counters each. In practice, this reduces the memory requirements by more than two orders of magnitude. Unfortunately, if implemented naively, it also increases the work to $\Theta(\binom{n}{2}NK)$ as we consider all K thinning counters for each of the N time steps and for each of the $\binom{n}{2}$ node pairs.
- Exploiting that our experiments consider very sparse graphs, we reduce the time complexity to $\mathcal{O}(\binom{n}{2} + NKm)$, by only updating counters of the edges existing at a certain point in time. We implement this idea using three values per thinned sequence, namely `last_step_observed`, `num_times_observed`, `num_of_01_transitions` with obvious semantics. In the second phase, we recover the x_{ij} counters using basic arithmetic.

3.2 Generalisation to wedges

Pertaining to our research question on the appropriate scale of graph features, it is natural to ask whether larger structures increase Autocorrelation's (AC) precision. To this end, we extend the method from pairs of nodes to triples, and define $3\binom{n}{3}$ sequences $Z_{u,\{v,w\}}$ and set $z^{(t)}_{u,\{v,w\}} = 1$ iff the wedge (v, u, w) exists at time t.[2] Then, we again consider their thinned subsequences in the obvious fashion. Based on the previously discussed algorithmic ideas, this leads to a memory usage of $\mathcal{O}(K\binom{n}{3})$, and a time complexity of $\mathcal{O}(K\binom{n}{3} + NK\sum_i \binom{d_i}{2})$ where $\sum_i \binom{d_i}{2}$ bounds the number of wedges (d_i being the degree of node i).

It is worth noting that the memory footprint can be trivially reduced by recording only a subset of the sequences. However, we consider this route to be outside the scope of the present work, as it is likely to introduce inaccuracies that would complicate a systematic comparison with the standard edge-based AC approach.

4 Mixing-Time as a classification task

The previously discussed Autocorrelation method has two limitations in regard to our research questions. Firstly, AC is limited to small local structures, such as edges and wedges, as scaling up to larger features is prohibitively expensive. Secondly, while the BIC approach

[1] `https://www.sandia.gov/app/uploads/sites/203/2022/06/rel01.zip`
[2] A wedge is a 2-path, i. e., we query whether node u is connected to both v and w.

has rigorous justification, we cannot verify that a particular sequence is correctly classified. To overcome these limitations, we introduce the novel ML-based approach Claim[3]. By design, it provides inherent ground-truth values to verify classification. It also permits broad exploration of a suite of arbitrary graph features, and subsequent systemic assessment of their influence on classifier performance.

Our framework shares similarities with *generative adversarial networks* (GANs) [33], where two models are trained conjointly: a generator Gen to mimic a training set T, and a discriminator Dis to tell instances from T and Gen apart. The intuition is that of a zero-sum game, i. e., a successful Dis suggests that the generator Gen emits recognisable artefacts. Thus, Gen needs to "learn" to produce more realistic instances by hiding these artefacts.

Claim also takes inspiration from Bläsius et al. [11] who propose a framework to systematically evaluate established network models. In this context and keeping our notation, Gen is a classic graph generator (using fitted parameters), while the discriminator Dis is trained to differentiate between generated and observed graphs. Crucially, the authors study the selection and importance of features that Dis uses. These give an interpretable feedback on the weaknesses of a graph model (e. g. that $G(n,p)$ fails to reproduce degree distributions), which may inspire improved models – in some sense manually closing the GAN loop. Instead of graph generators, Claim targets iterative graph randomisation schemes. The method can be more explicitly defined as follows.

Given an input graph G_0, we run the MCMC process to be studied for i steps to produce G_i, where $i \in \{\ell, f\}$ for parameters $0 < \ell \lll f$. Then, given the pair of graphs (G_0, G_i), the binary classification task is to decide whether $i = \ell$ or $i = f$. If the classifier Dis performs (slightly) better than a random coin flip, then (G_0, G_ℓ) and (G_0, G_f) must come from different distributions. Since we know that the MCMC converges to the uniform distribution after its mixing time τ, this clearly attests that ℓ steps do not suffice to reach this state.

We thus witness an empirical lower bound ℓ of the mixing time τ. Observe that this argument holds even if G_0 and G_f are not uncorrelated (i. e., $f < \tau$). Still, in order to ease the interpretation of our empirical results, we choose f sufficiently large to allow the assumption that G_0 and G_f are practically uncorrelated. In our experiments, we carry out a p-test against the hypothesis that the classifier's accuracy is governed by a binomial distribution. *Correct classification*, i. e., rejection of this hypothesis, *attests insufficient mixing.*

4.1 Interpretable features and classification

Claim is intended to give systematic feedback on the practical mixing of graphs, similar in spirit with the result of Bläsius et al. [11]. For this reason, we utilise established, interpretable graph features to describe the pair (G_0, G_i). This is an experimental design choice, not a technical limitation. We generated a suite of potentially discriminatory features:

- One scalar per graph: Degree Assortativity [55], Number of Connected Components
- Centralities: Betweenness [27], Closeness [28], Eigenvector [14], Katz [43], KPath [1], Laplacian [57]
- Other node scores: Core Decomposition [49], Local Clustering Coefficient [67], Local Square Clustering Coefficient [47], ORCA G5[4] [40], Common neighbours in G_0 and G_i.

[3] Correct Classification Attests Insufficient Mixing

[4] ORCA counts, for each node in the graph, its occurrences in each induced graphlet (all possible subgraphs of 4 and 5 nodes), as well as its occurrences in specific positions (known as orbits) within those graphlets.

As providing raw per-node features requires excessive memory[5], we compute summary statistics: mean, median, min, max, standard deviation, and a range of quantiles. As an additional benefit, the shape of the training set becomes invariant to the number of nodes n, which potentially allows training across graph sizes.

Classifier. To narrow the scope of the already excessive parameter range of our experiments, we focus on a single classification model; XGBoost [19]. Preliminary investigations indicated that more complex architectures, such as deep neural networks, offer only negligible performance gains at significantly increased computational cost. XGBoost is a supervised decision tree ensemble that utilises tree pruning and regularisation to mitigate overfitting and improve generalisation. We employ the `xgboost` package [19] with a binary cross-entropy loss function and regularisation penalties detailed in [19]. Crucially, this method allows us to quantify individual feature importances, a capability central to our research question.

5 Statistical tests

Claim is intentionally designed to provide ground truth labels to detect incorrect classifications, thereby addressing a key limitation of the Autocorrelation approach. However, while this ground truth is essential for the validation of our technique, it shifts the methodological uncertainty surrounding our research questions to a different locus: the efficacy of Claim depends critically on the classifier's ability to detect residual structure in a noisy data set. To address this concern, we adopt a two-pronged strategy: To begin, we carried out a number of precursory investigations to provide empirical evidence that XGBoost performs on-par with alternative classification models (cf. Section 4.1). We consequently consider it a good and representative choice among established classification methods. We complement this approach (although guided by feature importance results from trained classifiers) by statistically analysing the distribution of specific features as mixing progresses. We do this by testing the null hypothesis that the distribution of a feature at time t is identical to that of uncorrelated graph pairs; rejection indicates residual dependence on G_0. We primarily utilise the *two-sample z-test* to assess equality of population means. Given our large sample size (10^5), the central limit theorem ensures that the sampling distribution is approximately normal, providing a robust metric for detecting shifts in feature averages. We also performed Welch's t-test, Mann–Whitney U, Kolmogorov–Smirnov, and χ^2 tests, confirming that our conclusions are consistent across different distribution characteristics.

6 Empirical evaluation

The following section is divided into two parts. We first study the empirical methods themselves. Then, we apply the techniques to the MCMC processes *Edge Switching (ES)*, *Curveball (CB)*, and *Global Curveball (GCB)* to simple undirected graphs (see Section 2.2).

The total of measurements presented here required compute on the order of 10^5 core hours. Hence, a central concern of the experimental setup is to find trade-offs between the breadth, accuracy, and cost of the campaign.

[5] In a feature matrix, where each feature is encoded by a column and each row encodes the features of a specific instance, these features would in total yield $\approx 110n$ columns. Thus, at least 10^5 columns for most experiments.

Input instances. To ease comparison of our methods, we fix a set of degree sequences for all experiments. Concretely, we consider two common types: First, regular graphs, denoted by $\textsc{Reg}(n, d)$, where all n nodes have degree d. Secondly, graphs following a power-law degree distribution of exponent $2 \leq \gamma \leq 3$ and denoted by $\textsc{Pwl}(n, \gamma)$, where n again refers to the number of nodes.[6] Each run independently samples its start graph G_0 from $\mathcal{G}(\mathcal{D})$.

Arguably, starting and ending the chain in the same distribution imposes an additional burden on the measurements. For instance, in a precursor study, we used initial graphs sampled from the Random Hyperbolic Graph (RHG) [44, 12] model. This graph class is known to exhibit structurally distinctive characteristics, such as high clustering. Claim classifiers thus trivially focus on their gradients. However, these features quickly vanish for relatively short MCMC runs, and do not qualitatively affect the empirical lower bounds obtained. Accordingly, we randomly sample the start graphs from $\mathcal{G}(\mathcal{D})$, as this choice reduces initial transient behaviours in the measurements and, thereby, simplifies the interpretation of the results. Further, uniform sampling guarantees well-defined and comparable initial conditions across repetitions.

Unit step. The chains under consideration vastly differ in the degree of perturbation they inflict per transition. For instance, in a single step, ES affects at most 4 edges, whereas for certain graphs, GCB may alter a constant fraction of edges. Consequently, the computational cost per transition varies significantly across the chains and implementations. Since performance engineering considerations are beyond the scope of this work, we introduce a *unit step* as a runtime-agnostic progress measure. A unit step is defined such that each chain accesses every underlying element (nodes for CB and GCB, and edges for ES) in expectation exactly once:[7]

$$1 \text{ unit step} \equiv \frac{m}{2} \text{ steps of ES}, \quad \frac{n}{2} \text{ steps in CB}, \quad 1 \text{ step in GCB}$$

We emphasise that this definition is intended only to move plots of different chains into a comparable order of magnitude. There is no expectation that a unit step incurs the same work across chains, or that it induces the same degree of randomisation. After all, this non-trivial comparison between chains is one of the questions motivating this work.

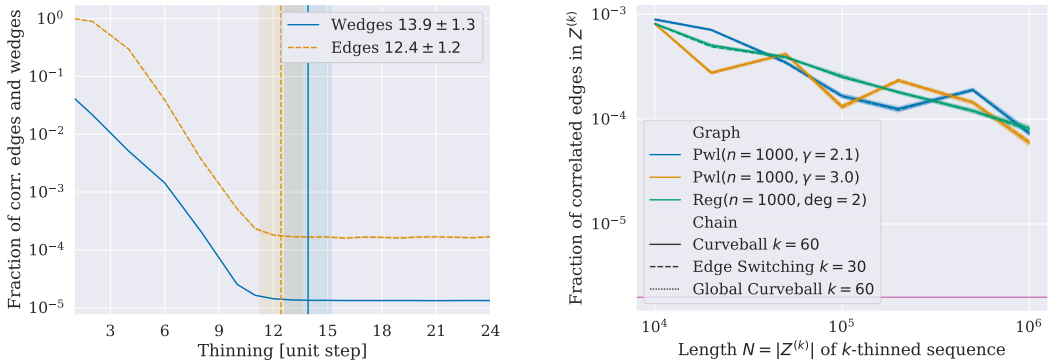
6.1 Performance of the methods

We first consider the methods in isolation mostly on $\textsc{Pwl}(n{=}1000, \gamma{=}2.1)$ – which is a difficult instance for (G)CB (cf., Section 6.2), and still satisfies $\gamma > 2.0$.[8]

6.1.1 Autocorrelation

We implement the autocorrelation (AC) method as discussed in Section 3.1 in the high-performance language Rust. The granularity of the underlying binary sequences is one unit step, or in other words, a thinning of k corresponds to one snapshot every k unit steps.

[6] To reduce variance, we sample 100 degree sequences with the requested parameters per graph. We then estimate their actual degree distribution parameters using [4], and select the closest match.
[7] Similar concepts have been used before; e. g. [18] define a so-called *super-step* using m steps of ES.
[8] Sequences with $\gamma \leq 2.0$ are called "anomalous" as even the average degree diverges. [8, Sec. 4.7]



(a) Convergence of AC for edges and wedges for Pwl(n=1000, γ=2.1). Vertical line indicates $\hat{\tau}$.

(b) Plateau value μ_p as function of sequence length. Horizontal line indicates resolution limit of $1/\binom{n}{2}$.

Figure 2 Fraction of correlated (w)edges as function of thinning (left) or sequence length (right).

Estimating mixing time $\hat{\tau}$. For an AC run r, let $f^{(r)}(k)$ be the fraction of edges deemed correlated as a function of the thinning value k (see Section 3 for details). Up to random noise $f^{(r)}(k)$ approaches a plateau monotonically for growing k (see e. g. Figure 2a)

To estimate a lower bound $\hat{\tau}$ of the mixing time, we adopt the framework used in [18] with increased precision. In the following, we consider a fixed parameter set (e. g. chain, degree sequence, run length). We carry out 50 independent runs $\{r_1, \ldots, r_{50}\}$ with sufficiently large thinning values $k_{\max}$ ensuring that the plateau is stable for at least 10 thinnings. Let μ_p and σ_p be the mean and, respectively, standard deviation of the plateau values of all runs.

For each run r, we then individually find the smallest thinning τ_r, such that $f^{(r)}(\tau_r)$ and its four next larger thinning steps have at most a distance of $2\sigma_p$ from μ_p. Finally, we report $\hat{\tau} = \mu_\tau \pm \sigma_\tau$ where μ_τ and σ_τ are the mean and standard deviation of $(\tau_1, \ldots, \tau_{50})$.

For ES, we consider the thinning values $\{1, \ldots, k_{\max}\}$, for $k_{\max} = 30$. For CB and GCB, we observe later convergence and, hence, set $k_{\max} = 60$. To reduce computational cost, we skip odd thinning values larger than 1. Hence, σ_τ tends to be smaller for ES.

Length of binary sequence N. A key parameter of AC is the number N of snapshots recorded per node pair (i. e., $N = |Z_{\{u,v\}}|$). To minimise work (scaling linearly in N), the parameter should be as small as possible. However, it may not be too small to offer sufficient significance: AC considers all $\binom{n}{2}$ node pairs and, for each time step, records the existence of all m edges. Hence, to observe at least one edge for each node pair, we need $N \cdot m \gg \binom{n}{2}$.

We select $N = 10^5$ (except in Figure 2b). This corresponds to an average of at least 50 observations per node pair and run for all considered instances with $n \leq 1000$. It is also significantly larger than previous works; e. g. [59, 18] seem to record less than 10^4 unit steps and then apply thinning. Thus, larger thinning values k have the shorter sequences.

In contrast, we use the same length for all sequences irrespective of the concrete choice of the thinning parameter (i. e., $|Z^{(k)}_{\{u,v\}}| = N$ for all k). This choice is motivated by the non-trivial interactions between the plateau value $f(k_{\max})$ and N as illustrated in Figure 2b. The general trend, an inverse relationship between $f(k_{\max})$ and N, can be attributed to the definition of the BIC. Nonetheless, the complex interactions suggest that the plateau can become ill-defined if the differently thinned subsequences had a different length.

Lastly, we observe that even larger values of N increase precision in some cases. As an extreme example, we see an increase from $\hat{\tau} = 40.2 \pm 2.2$ for $N = 10^5$ to $\hat{\tau} = 47.8 \pm 1.4$ for $N = 10^6$ for Pwl(n=1000, γ=2.1) and CB. However, we consider the 9.99× larger runtime unnecessary, as no qualitative differences could be observed.

Wedges. In Section 3.2 we ask whether AC can be improved by considering wedges (i. e., 2-path) instead of edges. Figure 2a supports this hypothesis, indicating a small but statistically significant improvement of $\hat{\tau}$. Unfortunately, we observe a 24.3-fold increase in runtime for PWL(n=1000, γ=2.1) when considering wedges. Even worse, the $\Theta(n^3)$ memory requirements translates to 122 GB of RAM in practice. This severely limits the number of parallel jobs that can be executed on a typical compute server. Combining both effects on our 32 core machine with 512 GB RAM, we observe a total reduction of throughput by a factor of 194.[9] Considering the small improvements observed, we deem this extension as not worth keeping.

6.1.2 CLAIM

We implement CLAIM in Python using, amongst others, the libraries `networkit` [6] for graph randomisation and analysis, `polars`[10] for data handling, as well as `scikit-learn` [56] and `xgboost` [19] for classification. As discussed in Section 4, we consider the randomisation steps ℓ (analogously to thinning in autocorrelation) and degree sequences in isolation.

Data generation and classification. For each degree sequence and randomisation step ℓ, we generate a balanced dataset of "correlated" instances (G_0, G_ℓ) and "uncorrelated" instances (G_0, G_f), where $f = 1000 \gg \ell$. To reduce computational cost, we reuse "uncorrelated" instances for different values of ℓ. Thus, for two differing values of ℓ the set of instances that represent G_f is the same. Since any classifier only has access to the instances associated with a fixed ℓ, this does not result in any advantage for the classifier. Performance is evaluated via 5-fold cross-validation. Since the classes are balanced, 50% accuracy represents random guessing. We consider a classifier as failed if a binomial test fails to reject the null hypothesis that it's success rate arises from a binomial process (with a confidence value of 0.01).

To reduce the effect of random noise (e. g. introduced by the dataset selection or during training), we repeat each training step 10 times with partially overlapping datasets. Combined with the 5-fold cross-validation, each data point is associated with 50 classifiers. We report $\hat{\tau} = (\ell_1 + \ell_2)/2$ as the mean between the mixing step ℓ_1 where at least 40 classifiers appear non-random and the first time step ℓ_2, where at least 40 classifiers appear random.

Dataset size. Unless otherwise specified, experiments utilise 10^5 pairs (5×10^4 pairs (G_0, G_l) and (G_0, G_f), each). To estimate the effect of this choice, we trained classifiers for PWL(n=1000, γ=2.1) on up to 10^6 pairs per class – increasing runtime and storage requirements by at least a factor of 20. This leads to only marginal performance gains with no new qualitative insights. Thus, in the interest of reproducibility, we use the smaller datasets for the following evaluation, although we emphasise that small gains in performance are possible by increasing the dataset size.

Feature relevance. Recall that our primary research questions pertain to the nature of graph features, thereby highlighting bottlenecks in FDSM MCMC processes. Hence, instead of more recent techniques, we rely on `xgboost` not only due to its performance, but also for its ability to quantify the relevance of individual features in the decision making process.

[9] It is straight-forward to parallelise the AC overhead in our implementation. However, given that the computation is already memory-bound, we expect diminishing returns without algorithmic improvements.
[10] `https://pola.rs`

By far, the most discriminatory feature picked up by `xgboost` is "common neighbours", i. e., the number of common neighbours shared by each node v between (G_0, G_ℓ) and (G_0, G_f) respectively. All other features have at least one order of magnitude smaller importance scores. Still, when considering the union of the three most relevant features of all successful classifiers ($1 \leq \ell \leq 17$) for ES on $\textsc{Pwl}(n{=}1000, \gamma{=}2.1)$, we observe 28 columns (recall that the distribution features are reduced to a number of descriptive statistics columns); of these, 22 represent the highly local features "common neighbours", ORCA, and degree assortativity. The remaining columns have such a low relevance score that we consider them insignificant.

The discriminatory power of "common neighbours" is interesting for two reasons: It is clear that this feature is in some sense a weakened version of the method that autocorrelation relies on. Secondly, it seems to indicate that local features take precedence over high-level graph attributes, despite the fact that the classifier has access to both. One could argue that the classifier performs well solely because it has access to both local *and* global features. When excluding the "common neighbours" metric, the performance of the classifier breaks down – almost halving the predicted value for $\hat{\tau}$ of CB and GCB for $\textsc{Pwl}(n{=}1000, \gamma{=}2.1)$. Still, this argument only shows that local features are necessary, and are *possibly* acting in concert with global features. We uncover evidence to the contrary using statistical tests.

Statistical tests. We aim to provide evidence that global features do not provide information beyond what is captured by the "common neighbours" metric. To this end, we show that statistical tests can distinguish the distributions of (G_0, G_ℓ) and (G_0, G_f) solely based on the realisation of the "common neighbours" feature up to values of ℓ where Claim breaks down.

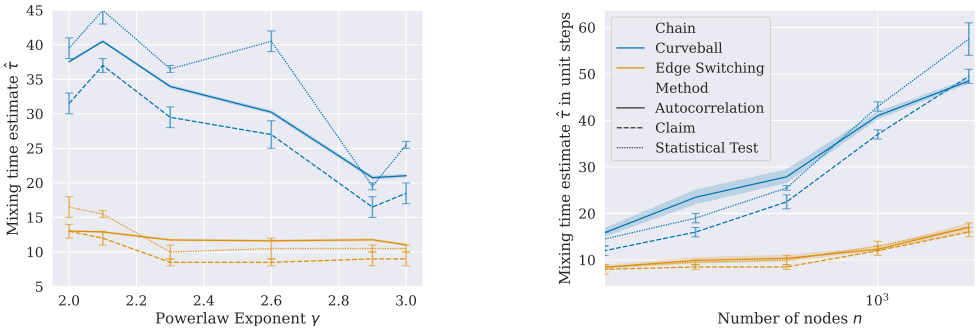
The tests are two-sided and conducted at a 95% confidence level ($\alpha = 0.05$). We carried out all tests listed in Section 5, yielding similar findings across the board. As such, we focus only on z-test results, as they generally exhibit the best discriminatory ability. Indeed, as seen in Figure 3a, the classifier performance can be matched or even surpassed by statistical tests on the "common neighbours" feature alone. We attribute the increased performance partly to the difference in scale – the classifier decides on an individual pair of graphs without global knowledge, while the statistical tests can leverage the global distribution.

However, it is clear that the classifier fails to gain a tangible advantage by having access to various global and local graph features beyond what is provided by the highly local "common neighbours" feature. This lends further credence to the evidence that local features outrank the provided global features in discriminatory power for converging graph distributions.

6.2 Performance of MCMC processes

In the following, we shift the focus to the three MCMC methods *Edge Switching (ES)*, *Curveball (CB)*, and *Global Curveball (GCB)*. We use Autocorrelation (AC), Claim, and statistical tests (ST) to quantify the chains' performances. For each method, we use the parameters established in the previous section. Recall that each parameter represents a trade-off between precision, robustness, and computational effort – hence bounds can usually be influenced, e. g. by changing the data set size. Thus, in the following, we will not compare the absolute bounds each method provides, but rather whether they agree qualitatively.

Across parameter investigations, we see broad agreement between the ST and Claim results. Absolute bounds are marginally different, but the qualitative agreement between these approaches suggests substantive findings, particularly given the contrast between these and the AC results in certain parameter regimes.



(a) Scaling $2 \leq \gamma \leq 3$ for $\textsc{Pwl}(n{=}1000, \gamma)$.

(b) Scaling $125 \leq n \leq 2000$ for $\textsc{Pwl}(n, \gamma{=}2.1)$.

Figure 3 Mixing time estimates $\hat{\tau}$ as function of nodes and power-law exponent. For AC we report the 95% confidence interval. For Claim and ST the errorbars indicate the spread between the last point where at least 80% of repeats indicated correlated data and the first point where at most 20% do. Both plots share the same legend.

Regular graphs. Regular graphs are an easy input class for all MCMC processes studied: ES exhibits no significant rejection rate (see Table 3), i. e., most switches modify two edges. Similarly, CB exchanges yield, as expected, $d/2$ neighbours per trade (i. e., modifies d edges). Thus, both chains modify almost m edges per unit step; however some are repeatedly sampled. If we ignore dependencies within a step (as $d \lll m$), the situation is analogous to the Coupon collector problem [54], predicting that $(1 - 1/e)m \approx 0.63m$ edges are updated per unit step – with no additional dependency on the degree d. Our measurements in Table 3 support this.

Consistently, none of our methods suggest a trend in the mixing time estimates when scaling the degree of $\textsc{Reg}(n{=}1000, d)$ in the range $2 \leq d \leq 6$. AC finds a median (among different values of d) estimate of 10 for ES and CB, and a slightly faster mixing von GCB of 8. Claim and ST yield identical trends with one to two steps higher estimates. The better performance of GCB can, at least in part, be explained by the same logic as used previously: as CB draws with replacement, only $\approx 63\%$ of nodes are considered per unit step. In contrast, GCB samples without replacement, and thereby considers all nodes. Hence, almost all edges are considered twice per unit step (once for each end point), and each is modified with a 50% probability per trade. In other words, the probability that an edge is updated after being considered (approximately) twice per unit step, tends to 3/4. This is consistent with the measurements reported in Table 3, showing that GCB updates 1.18 times more edges in a unit step than ES and CB.

Effect of the power law exponent. To study the effect of inhomogeneous degree sequences, we switch to power law distributions with parametrisable skewness $2 \leq \gamma \leq 3$. The smaller γ, the more high degree nodes we observe; more specifically, the expected maximum degree scales as $\Theta\left(n^{1/(\gamma-1)}\right)$ as $n \to \infty$. [8, Sec 4.3]

Figure 3a summarises the mixing time estimations of the three methods for ES and CB. Since Curveball and its global variant perform similarly (with a slight advantage for GCB, consistent with our discussion on regular graphs), we omit GCB for clarity of the presentation. With the exception of $\textsc{Pwl}(n{=}1000, \gamma)$ with $\gamma \leq 2.1$, Edge Switching is largely unaffected by γ. The peak at low values of γ can be largely attributed to the fact that ES's rejection rate scales in the degrees of the edges' endpoints, relative to the total number of edges. As indicated in Table 2, the maximum degree of $\textsc{Pwl}(n{=}1000, \gamma{=}2.0)$ is 236, leading to an

acceptance rate of only 39%. Already for PWL(n=1000, γ=2.3), the rejection rate decreases to 7 %, and further to below 1% for PWL(n=1000, γ=3.0). This coincides with the plateau of the mixing time estimates $\hat{\tau}$ for $\gamma \geq 2.3$.

Interestingly, we observe a qualitative difference between ES and CB: while ES is almost unaffected by the value of γ, the estimates for CB's mixing time halve as $\gamma \to 3$. We are unaware of an empirical demonstration of this effect in the literature.

Graph size. In practice, graph size is a crucial parameter for the mixing time. In Figure 3b, we summarise the dependency between the number of nodes n and the mixing time estimates $\hat{\tau}$. We again omit GCB for clarity, as it is qualitatively similar to CB (barring an outlier for $n = 2000$; c.f. Figure 4). Observe that a unit step already scales linearly in the graph size, thus the figure is consistent with $\tau = \Omega(n \log n)$. This is plausible because CB needs – analogously to the Coupon collector – expected $\Omega(n \log n)$ trades until each edge could be traded once. Similarly, ES requires expected $\Omega(m \log m)$ switches. These findings contradict practical folklore which suggests to select the number of steps as a constant factor of the graph size.

Figure 3b also indicates a more pronounced response from CB. This is despite the fact that the fraction of nodes with degree 1 is reduced from 0.9 to 0.68 for PWL(n=125, γ=2.1) and PWL(n=1000, γ=2.1), respectively. At the same time, the average degree grows from 1.86 to 3.0; both effects increase the efficiency of CB. The widening performance gap between ES and CB can be partly attributed to the scaling of the plot. Recall that a unit step is defined as $m/2$. Hence, the increase in the average degree implicitly leads to a higher number of edge switches per node, even if $\hat{\tau}$, expressed in unit steps, remains constant.

7 Conclusion and outlook

We considered several methods to empirically quantify the mixing time of graph randomisation processes. The methods agree qualitatively on virtually all instances despite their different methodological foundations and disjoint code bases. This suggests that the results are robust.

The AC method remains highly relevant as a favourable compromise between performance and accuracy. Our novel ML-based approach CLAIM gives insights into which features best reveal residual structures in incompletely mixed instances. These are predominately local in nature and include "common neighbours", ORCA counts and degree assortativity. We support this observation with statistical tests based solely on a local feature. Nonetheless, our experiments including higher order local structures such as wedges and small subgraphs captured by ORCA counts suggest that there is a delicate trade-off between the information conveyed by larger substructures and the increasing difficulty to separate the aggregated noise from a genuine signal.

We demonstrate improved "empirical lower bounds" for power law sequences with low exponent. Together with the observed super-linear scaling in the graph size, this suggests that – especially for larger networks – significantly more than $10m$ switches and more than $50n$ trades should be used. Our results suggest that the ES chain is less susceptible to skewed degree distributions than Curveball. For near regular graphs, however, we find very similar behaviours for all chains. In this regime GCB seems favourable, as it shows the lowest mixing time bounds, and engineered implementations [18] significantly outperform ES.

As an orthogonal concern, it remains open whether more sophisticated classifiers (e. g. graph neural networks) with less interpretable features may boost CLAIM sensitivity.

References

1 Tharaka Alahakoon, Rahul Tripathi, Nicolas Kourtellis, Ramanuja Simha, and Adriana Iamnitchi. K-path centrality: a new centrality measure in social networks. In *SNS*. ACM, 2011. doi:10.1145/1989656.1989657.

2 Daniel Allendorf, Ulrich Meyer, Manuel Penschuck, and Hung Tran. Parallel global edge switching for the uniform sampling of simple graphs with prescribed degrees. *J. Parallel Distrib. Comput.*, 2023.

3 Daniel Allendorf, Ulrich Meyer, Manuel Penschuck, Hung Tran, and Nick Wormald. Engineering uniform sampling of graphs with a prescribed power-law degree sequence. In *ALENEX*. SIAM, 2022. doi:10.1137/1.9781611977042.3.

4 Jeff Alstott, Ed Bullmore, and Dietmar Plenz. `powerlaw`: a Python package for analysis of heavy-tailed distributions. *PloS one*, 9(1):e85777, 2014.

5 Georgios Amanatidis and Pieter Kleer. Rapid mixing of the switch Markov Chain for strongly stable degree sequences. *RSA*, 2020.

6 Eugenio Angriman, Alexander van der Grinten, Michael Hamann, Henning Meyerhenke, and Manuel Penschuck. Algorithms for large-scale network analysis and the networkit toolkit. In *Algorithms for Big Data*, Lecture Notes in Computer Science, pages 3–20. Springer, 2022. doi:10.1007/978-3-031-21534-6_1.

7 Andrii Arman, Pu Gao, and Nicholas C. Wormald. Fast uniform generation of random graphs with given degree sequences. *Random Struct. Algorithms*, 2021. doi:10.1002/RSA.21004.

8 Albert-László Barabási. *Network science book*. Cambridge University Press Cambridge, 2014.

9 Edward A. Bender and E. Rodney Canfield. The asymptotic number of labeled graphs with given degree sequences. *J. Comb. Theory A*, 1978. doi:10.1016/0097-3165(78)90059-6.

10 Annabell Berger and Corrie Jacobien Carstens. Smaller universes for uniform sampling of {0,1}-matrices with fixed row and column sums. *CoRR*, 2018.

11 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Anton Krohmer, and Jonathan Striebel. Towards a systematic evaluation of generative network models. In *WAW*, LNCS. Springer, 2018. doi:10.1007/978-3-319-92871-5_8.

12 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Ulrich Meyer, Manuel Penschuck, and Christopher Weyand. Efficiently generating geometric inhomogeneous and hyperbolic random graphs. *Netw. Sci.*, 2022. doi:10.1017/NWS.2022.32.

13 Bela Bollobás. *Random graphs*. Academic Press, 1985.

14 Phillip Bonacich. Technique for analyzing overlapping memberships. *Sociological meth.*, 1972.

15 C. J. Carstens. Motifs in directed acyclic networks. In *SITIS*. IEEE Computer Society, 2013. doi:10.1109/SITIS.2013.99.

16 Corrie Jacobien Carstens. *Topology of complex networks: models and analysis*. PhD thesis, RMIT, 2016.

17 Corrie Jacobien Carstens, Annabell Berger, and Giovanni Strona. A unifying framework for fast randomization of ecological networks with fixed (node) degrees. *MethodsX*, 2018.

18 Corrie Jacobien Carstens, Michael Hamann, Ulrich Meyer, Manuel Penschuck, Hung Tran, and Dorothea Wagner. Parallel and I/O-efficient randomisation of massive networks using global curveball trades. In *ESA*, 2018.

19 Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2939672.2939785.

20 Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proc. Natl. Acad. Sci.*, 2002. doi:10.1073/pnas.252631999.

21 Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Ann. Comb.*, 2002. doi:10.1007/PL00012580.

22 C. Cooper, M. E. Dyer, and C. S. Greenhill. Sampling regular graphs and a peer-to-peer network. *Comb. Probab. Comput.*, 2007. doi:10.1017/S0963548306007978.

23 Péter L. Erdös, Catherine S. Greenhill, Tamás Róbert Mezei, István Miklós, Daniel Soltész, and Lajos Soukup. The mixing time of switch markov chains: A unified approach. *Eur. J. Comb.*, 2022. doi:10.1016/j.ejc.2021.103421.

24 Péter L. Erdős, Sándor Z. Kiss, István Miklós, and Lajos Soukup. Approximate counting of graphical realizations. *PloS one*, 2015. doi:10.1371/journal.pone.0131300.

25 Max Espinoza. On network randomization methods: A negative control study. *Fairfield, CT: Fairfield University.*, 2012.

26 Jacob G Foster, David V Foster, Peter Grassberger, and Maya Paczuski. Edge direction and the structure of networks. *Proceedings of the National Academy of Sciences*, 107(24):10815–10820, 2010.

27 Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 1977.

28 Linton C Freeman. Conceptual clarification. *Soc. Net.: Critical Concepts in Sociology*, 2002.

29 Pu Gao and Catherine S. Greenhill. Mixing time of the switch Markov Chain and stable degree sequences. *Discret. Appl. Math.*, 2021. doi:10.1016/J.DAM.2020.12.004.

30 Pu Gao and Nicholas C. Wormald. Uniform generation of random regular graphs. *SIAM J. Comp.*, 2017.

31 Pu Gao and Nicholas C. Wormald. Uniform generation of random graphs with power-law degree sequences. In *SODA*. SIAM, 2018. doi:10.1137/1.9781611975031.114.

32 Christos Gkantsidis, Milena Mihail, and Ellen W. Zegura. The Markov Chain simulation method for generating connected power law random graphs. In *ALENEX*, 2003.

33 Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

34 Nicholas Gotelli and Gary Graves. *Null models in ecology*. Smithsonian Institution, 1996.

35 C. S. Greenhill. The switch Markov Chain for sampling irregular graphs. In *SODA*, 2015. doi:10.1137/1.9781611973730.103.

36 Catherine S. Greenhill and Matteo Sfragara. The switch Markov Chain for sampling irregular graphs and digraphs. *TCS*, 2018.

37 S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *J. SIAM*, 1962. doi:10.1137/0110037.

38 Michael Hamann, Ulrich Meyer, Manuel Penschuck, Hung Tran, and Dorothea Wagner. I/O-efficient generation of massive graphs following the *LFR* benchmark. *ACM J. Exp. Algorithmics*, 2018. doi:10.1145/3230743.

39 Václav Havel. Poznámka o existenci konečných grafů. *Časopis pro pěstování matematiky*, 1955. URL: http://eudml.org/doc/19050.

40 Tomaž Hočevar and Janez Demšar. Combinatorial algorithm for counting small induced graphs and orbits. *PloS one*, 2017.

41 M. Jerrum and A. Sinclair. Fast uniform generation of regular graphs. *TCS*, 1990. doi:10.1016/0304-3975(90)90164-D.

42 R. Kannan, P. Tetali, and S. S. Vempala. Simple Markov Chain algorithms for generating bipartite graphs and tournaments. *RSA*, 1999.

43 Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 1953.

44 Dmitri V. Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *CoRR*, abs/1006.5169, 2010. arXiv:1006.5169.

45 Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, 2009.

46 Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 2008.

47 Pedro G Lind, Marta C Gonzalez, and Hans J Herrmann. Cycles and clustering in bipartite networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 2005.

48 P. Mahadevan, D. V. Krioukov, K. R. Fall, and V. Vahdat. Systematic topology analysis and generation using degree correlations. In *SIGCOMM*, 2006.

49 Fragkiskos D. Malliaros, Christos Giatsidis, Apostolos N. Papadopoulos, and Michalis Vazirgiannis. The core decomposition of networks: theory, algorithms and applications. *VLDB J.*, 2020. doi:10.1007/S00778-019-00587-4.

50 B. D. McKay. Asymptotics for symmetric 0-1 matrices with prescribed row sums. *Ars Combinatoria*, 1985.

51 B. D. McKay and N. C. Wormald. Uniform generation of random regular graphs of moderate degree. *J. Algorithms*, 1990. doi:10.1016/0196-6774(90)90029-E.

52 R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 2002.

53 Ron Milo, Nadav Kashtan, Shalev Itzkovitz, Mark EJ Newman, and Uri Alon. On the uniform generation of random graphs with prescribed degree sequences. *arXiv preprint cond-mat/0312028*, 2003.

54 Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

55 Mark EJ Newman. Mixing patterns in networks. *Physical review E*, 2003.

56 Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. `scikit-learn`: Machine learning in Python. *J. ML. Res.*, 2011.

57 Xingqin Qi, Eddie Fuller, Qin Wu, Yezhou Wu, and Cun-Quan Zhang. Laplacian centrality: A new centrality measure for weighted networks. *Inf. Sci.*, 2012. doi:10.1016/J.INS.2011.12.027.

58 A Ramachandra Rao, Rabindranath Jana, and Suraj Bandyopadhyay. A markov chain monte carlo method for generating random (0, 1)-matrices with given marginals. *Sankhyā: The Indian J. of Stat., Series A*, 1996.

59 Jaideep Ray, Ali Pinar, and C. Seshadhri. A stopping criterion for markov chains when generating independent random graphs. *J. Complex Networks*, 2015. doi:10.1093/COMNET/CNU041.

60 Steffen Rechner and Annabell Berger. Marathon: an open source software library for the analysis of markov-chain monte carlo algorithms. *PloS one*, 11(1):e0147935, 2016.

61 Alan Sokal. Monte carlo methods in statistical mechanics: foundations and new algorithms. In *Functional integration: Basics and applications*. Springer, 1997.

62 Isabelle Stanton and Ali Pinar. Constructing and sampling graphs with a prescribed joint degree distribution. *ACM J. Exp. Algorithmics*, 2011. doi:10.1145/2133803.2330086.

63 Giovanni Strona, Domenico Nappo, Francesco Boccacci, Simone Fattorini, and Jesus San-Miguel-Ayanz. A fast and unbiased procedure to randomize ecological binary matrices with fixed row and column totals. *Nature communications*, 2014.

64 N. D. Verhelst. An efficient MCMC algorithm to sample binary matrices with fixed marginals. *Psychometrika*, 2008.

65 F. Viger and M. Latapy. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. *J. Complex Networks*, 2016. doi:10.1093/comnet/cnv013.

66 Guanyang Wang. A fast MCMC algorithm for the uniform sampling of binary matrices with fixed margins. *Electron. J. Stat.*, 2020. doi:10.1214/20-EJS1702.

67 Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *Nature*, 1998.

68 Nicholas C. Wormald. Generating random regular graphs. *J. Algorithms*, 1984. doi:10.1016/0196-6774(84)90030-0.

A Efficiency of MCMC processes

In this section, we report basic statistics on the efficiency of the MCMC processes, such as the number of edges that are different after a unit step.

Table 1 Efficiency of MCMC processes for powerlaw sequence of varying size. "Upd": Number of edges present only after randomization. "Unit": per unit step, "Trd": per CB trade. "AccRate": Acceptance rate of ES (i. e., fraction of successful swaps).

Instance					Edge Switch		Curveball		Global CB
			Degree		Acc-	Upd	Upd	Upd	Upd
Model	n	m	Min	Max	Rate	/Unit	/Trd	/Unit	/Unit
Pwl(γ=2.1)	125	116	1	26	0.79	61.6	1.22	52.5	63.2
Pwl(γ=2.1)	250	260	1	40	0.81	138	1.25	113	136
Pwl(γ=2.1)	500	582	1	71	0.83	320	1.40	246	294
Pwl(γ=2.1)	1000	1501	1	241	0.65	673	1.36	495	598
Pwl(γ=2.1)	2000	3657	1	542	0.52	1332	1.38	1018	1224

Table 2 Efficiency of MCMC processes for powerlaw sequence of varying exponent. "Upd": Number of edges present only after randomization. "Unit": per unit step, "Trd": per CB trade. "AccRate": Acceptance rate of ES (i. e., fraction of successful swaps).

Instance					Edge Switch		Curveball		Global CB
			Degree		Acc-	Upd	Upd	Upd	Upd
Model	n	m	Min	Max	Rate	/Unit	/Trd	/Unit	/Unit
Pwl(γ=2.0)	1000	1715	1	236	0.61	722	1.47	543	649
Pwl(γ=2.1)	1000	1501	1	241	0.65	673	1.36	495	598
Pwl(γ=2.3)	1000	1000	1	61	0.93	595	1.35	461	552
Pwl(γ=2.6)	1000	784	1	44	0.96	483	1.17	409	488
Pwl(γ=2.9)	1000	651	1	13	1.00	410	1.13	377	449
Pwl(γ=3.0)	1000	640	1	20	0.99	402	1.13	371	441

Table 3 Efficiency of MCMC processes for regular sequence of varying degree. "Upd": Number of edges present only after randomization. "Unit": per unit step, "Trd": per CB trade. "AccRate": Acceptance rate of ES (i. e., fraction of successful swaps).

Instance					Edge Switch		Curveball		Global CB
			Degree		Acc-	Upd	Upd	Upd	Upd
Model	n	m	Min	Max	Rate	/Unit	/Trd	/Unit	/Unit
Reg(deg =2)	1000	1000	2	2	1.00	630	2.04	633	749
Reg(deg =3)	1000	1500	3	3	0.99	943	3.06	946	1120
Reg(deg =4)	1000	2000	4	4	0.99	1257	4.01	1259	1495
Reg(deg =5)	1000	2500	5	5	0.99	1567	4.88	1574	1864
Reg(deg =6)	1000	3000	6	6	0.99	1878	6.00	1886	2234

B Summary plots

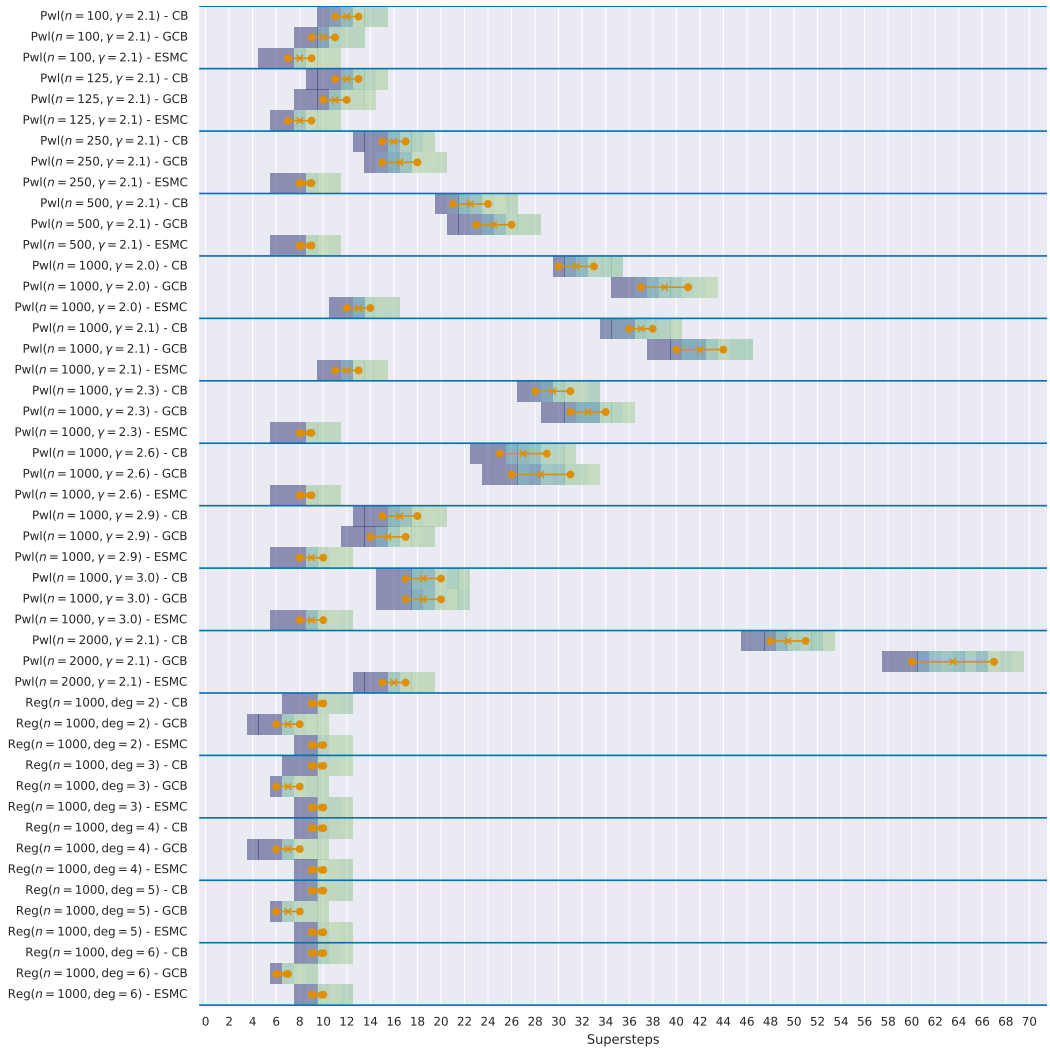


Figure 4 Summary of all CLAIM results. Recall that we trained 50 classifiers for each combination of graph instance, MCMC process, and randomisation length. The former two are combined on the Y-axis, the latter indicated on the X-axis. For each classifier, we carry out a binomial test to reject the hypothesis makes random decisions. The darker the heatmap's shade, the more classifiers are considered non-random. To find the position of the phase transition, we consider the right-most point where at least 40 classifiers appear non-random, and the left-most point where at least 40 classifiers appear random.