Increasing the Sampling Efficiency for the Link Assessment Problem

André Chinazzo(✉), Christian De Schryver, Katharina Zweig, and Norbert Wehn

TU Kaiserslautern, Kaiserslautern, Germany
{chinazzo,schryver,wehn}@eit.uni-kl.de, zweig@cs.uni-kl.de

Abstract. Complex graphs are at the heart of today's big data challenges like recommendation systems, customer behavior modeling, or incident detection systems. One reoccurring task in these fields is the extraction of network motifs, which are subgraphs that are reoccurring and statistically significant. To assess the statistical significance of their occurrence, the observed values in the real network need to be compared to their expected value in a random graph model.

In this chapter, we focus on the so-called Link Assessment (LA) problem, in particular for bipartite networks. Lacking closed-form solutions, we require stochastic Monte Carlo approaches that raise the challenge of finding appropriate metrics for quantifying the quality of results (QoR) together with suitable heuristics that stop the computation process if no further increase in quality is expected. We provide investigation results for three quality metrics and show that observing the right metrics reveals so-called *phase transitions* that can be used as a reliable basis for such heuristics. Finally, we propose a heuristic that has been evaluated with real-word datasets, providing a speedup of 15.4× over previous approaches.

Keywords: Link Assessment · Edge switching · Curveball · Random graphs

1 Introduction

The *data deluge* phenomenon is ever more present. We, as a society, generate and store far more data than what we can make use of right now [7]. Among the reasons for that are: 1. new approaches to acquire data, ranging from internet traffic recordings to high throughput DNA sequencing; and 2. the reduction in price per bit of data storage technologies, which motivates companies and researchers to be less selective about what data to be stored. These are by no means disadvantages over past methodologies, instead, they open new possibilities for data analysis that require methods that are more efficient and more robust against noise.

Complex network analysis is a tool-set of methods commonly used to extract information from large amounts of data, as long as the data can be meaningfully represented as a network. One popular method is the so-called *Link Assessment (LA)*, whose goal is to refine the data based on the principle of *structural similarity* (or *homophily*), i.e., entities that are alike tend to share a large proportion of their neighbors. Although the assumption of homophily is most common in social network analysis, mainly unipartite networks, it has been shown useful in a large range of contexts, including bipartite networks (such as a user rating / movie network) [24 SPP].

H. Bast et al. (Eds.): Algorithms for Big Data, LNCS 13201, pp. 39–56, 2022.
https://doi.org/10.1007/978-3-031-21534-6_3

For unipartite networks, such as a protein-protein interaction database [12] or social networks [24 SPP], the LA may serve as a data-cleansing method, evaluating whether each existing link is likely to be a true positive and whether each non-existing link is likely to be a true-negative. This should not be confused with the link *prediction* problem, which is already well-researched [17], but poses a slightly different question: Given a snapshot of a network at time t, which of the yet unconnected node pairs are predicted to be connected at $t + 1$?

For bipartite networks, such as genes that are associated with diseases [11] or products that are bought by costumers [10], the LA is a systematic way of projecting such networks to one of their sides [27]. This so-called *one-mode projection* transforms a bipartite network into a unipartite one by connecting the nodes on one of the sides based on their connections to the other side, while the nodes of the other side are discarded (see Sect. 2). Since most methods and tools for network analysis focus on general graphs, the one-mode projection of bipartite networks is a particularly useful pre-processing step to their analysis [27]. In this chapter, therefore, we focus on the LA for bipartite networks.

The LA is closely related to a vast body of research that includes the link prediction [17,20], recommendation systems [2], and node similarity in complex network analysis [16,18]. Known approaches for such problems can be divided into *supervised* and *unsupervised learning*. Supervised learning approaches require a ground truth, i.e., a subset of the network whose links or labels are known to be correct. In general, these ground truths, or training sets, are manually annotated and therefore are often the bottleneck in the data-mining pipeline [26]. Moreover, the ground truths are often split into a training set and a test set, where the test set is used to estimate the quality of results (QoR). Unsupervised learning methods, on the other hand, require no ground truth, instead, they rely only on the structure (or other properties) of the data itself. Thus, in many cases, the QoR cannot be directly estimated. In conclusion, these methods should only be applied to specific types of data set for which their robustness has already been validated.

As stated earlier, the LA is an unsupervised method that assumes the relationships in the network to adhere to the notion of homophily, where alike nodes tend to have a larger common neighborhoods than one would expected from merely their degrees in a randomly constructed graph. In practice, the LA is based on Markov chain Monte Carlo (MCMC) methods that generate a large set of such random graphs. These MCMC methods are known to eventually converge, but their parameters are unknown. Whenever a ground truth is available, the QoR is assessed by, for e.g., the ratio of correctly identified pairs of alike nodes over the total number of pairs listed in the ground truth, i.e., the PPV_k (see Sect. 2.1). Else, one must ensure that the MCMC has converged.

In this chapter we summarize specific aspects for creating an LA problem solver, give insights into available metrics for measuring the QoR, and propose an appropriate heuristic that can speed up the run time by a factor of 15.4× compared to a conservative approach.

2 Link Assessment Based on z*

Several node similarity measures have been proposed by different scientific communities, such as the *Jaccard index*, the *Pearson correlation coefficient*, or the *hypergeom* [12]. In [24 SPP], we have introduced a new similarity measure, the z* that has shown to be the most robust one across a range of datasets from protein-protein interactions to movie ratings to social network.

Given a bipartite graph $G((V_l, V_r), E)$ with vertices V_l and V_r and edges E, we define $coocc(u, v)$ as the number of co-occurring neighbors of nodes u and v. Most node similarity measures inherently depend on this quantity, but differ in how they are normalized based on the structure of the network. The similarity scores between nodes of the side-of-interest, say V_l, are the basis for the one-mode projection $G((V_l, V_r), E) \Rightarrow G'(V_l, E')$, where E' are edges between nodes in V_l. Some similarity measures use a simple factor based on properties of the two nodes, u and v (e.g., Jaccard index), while others are the result of a comparison to the expected value from a null-model (e.g., hypergeom).

The z* falls in the second category, as a combination of the p-value and the z-score statistics of the node-pairwise co-occurrences. Node pairs are ranked more similar if their p-value is smaller and ties are broken by their z-score [24 SPP]. Of key importance is the null-model used–the *fixed degree sequence model (FDSM)*. The FDSM is a random graph model that preserves the degree sequence of the original network while randomizing its nodes' interconnections, or edges. While it has been shown that the FDSM is a superior null-model than simpler graph models [13, 14, 27], closed-form expressions for the expected co-occurrences, $coocc_{FDSM}(u, v)$, are not known. These quantities are instead estimated by a random sampling procedure, known as a Markov chain Monte Carlo (MCMC) approach. Algorithm 1 describes the complete calculation of the z*.

2.1 Ground Truth and PPV_k

Throughout this chapter, we discuss a variety of results for the Link Assessment (LA) using as an example the Netflix Prize dataset[1]. By setting a threshold, the data are represented as a bipartite graph between users and movies, where an edge (u, v) means that user u liked (4 or 5 stars in the 1–5 scale) movie v. By finding significant co-occurrences between any two movies (v, w), a one-mode projection can be obtained [27]. The projection to the movies side was preferred because the users are anonymized, therefore it would be impossible to generate a ground truth of known similar users.

We quantify the quality of the LA by the positive predictive value (PPV_k) based on a ground truth dataset that contains only pairs of known non-random association, namely movie sequels like Star Wars and James Bond. The PPV_k is the fraction of correctly identified pairs from the ground truth in the set of the k highest ranked pairs of movies, where k is the number of pairs in the ground truth (see [3 SPP] for an example).

Building a ground truth for real datasets requires orthogonal information about the data (information that is not available for the LA method being tested) as well as an

[1] Available at https://www.kaggle.com/netflix-inc/netflix-prize-data.

Algorithm 1: The complete Link Assessment algorithm, calculating the similarity measure z*

Data: Graph $G((V_l, V_r); E)$ with vertices V_l and V_r and edges E, V_l being the vertices of interest;

Result: A z*-score (p-value and z-score) for all pairs of vertices $(u, v) \in (V_l \times V_l)$;

Calculate $coocc(u, v) \; \forall \; (u, v) \in (V_l \times V_l)$; $G_0 := G$;

for *i := 1 to* $|samples|$ **do**

  $G_i := G_{i-1}$;

  Graph randomization:

  for $|swaps|$ **do**

    Choose two edges at random in G_i and swap them, if no duplicate edge arises from the swap;

  Coocc computation:

  Calculate $coocc_i(u, v) \; \forall \; (u, v) \in (V_l \times V_l)$;

Calculate p-value and z-score, i.e., the z:*

$\text{p-value}(u, v) := (|\{i : coocc_i(u, v) > coocc(u, v) \; \forall \; i \in 1..|samples|)\}| \; \forall \; (u, v) \in (V_l \times V_l)$;

$coocc_{FDSM}(u, v) := \{coocc_i(u, v) \; \forall \; i \in 1..|samples|\} \; \forall \; (u, v) \in (V_l \times V_l)$;

$\text{z-score}(u, v) := \frac{mean(coocc_{FDSM}(u,v)) - coocc(u,v)}{stddev(coocc_{FDSM}(u,v))} \; \forall \; (u, v) \in (V_l \times V_l)$;

reliable method, such as assuming that movies within a sequel are non-randomly similar. Therefore, reliable ground truths are rare, limiting the range of input datasets for which the PPV_k can be measured.

Recently, however, we have discovered a systematic way of generating synthetic graphs for which the ground truth can be directly extracted, based on the benchmarks proposed in [14]. With that, we are able to conduct experiments for reliably comparing the efficiency and QoR of several LA approaches over an arbitrary range of input datasets. However, this work is still ongoing.

2.2 Random Graph Models

Network mofits are subgraphs whose occurrence in the observed data is statistically significant when compared to a random graph model (a null-model). The choice of such a null-model must be well-suited to test the investigator's hypothesis, and an inappropriate null-model can result in misinterpretation of the observed data [8]. The fixed degree sequence model (FDSM) is considered most appropriate for the identification of motifs, and in many cases, only simple graphs should be considered, i.e., no self-loops nor multi-edges. Unfortunately, closed-form expressions for the expected motif frequency over all possible simple graphs with a prescribed degree sequence are not yet known. Therefore, we commonly rely on a comparatively inefficient MCMC approach based on sequential mixing of the sampled graph states.

In [23 SPP] and [22 SPP], we have looked at different null-models, which can be more efficiently generated, as an approximation for the FDSM, as well as developed equations with the same intention. While some 3-node subgraph frequencies can be well approximated by simple equations, the case for the node-pairwise co-occurrences

is more complex. For very regular degree sequences, i.e., all nodes have similar degrees, an equation based on the simple independence model is sufficient to estimate the individual node pairs co-occurrences. As the degree sequence becomes more skewed, the true values from the FDSM diverge from the approximation. Even a more intricate approximation for the individual co-occurrences [19, p. 441], whose sum almost matches the true value, becomes inaccurate for high-degree nodes. Since skewed degree sequences are abundant in real networks, such approximations cannot be widely used.

2.3 Co-occurrence Gradient in the FDSM

In another attempt to avoid the costly MCMC sampling approach for estimating the expected co-occurrences in the FDSM, we have analyzed the co-occurrence gradients throughout the Markov chains–a so-called *mean-field approach*, borrowed from statistical physics. In this approach, we first find the differential equation that describes the expected change, i.e., gradient, in co-occurrence after one single step in the graph mixing Markov chain. If the gradient is sufficient to describe the dynamics of the chain, a closed-form solution for the expected co-occurrences could be derived (see [1] for an example of a successful attempt), or at least an iterative, direct method that is not based on sampling.

In order to fully describe the dynamics of the mixing chains, the co-occurrences gradient, $\Delta\mathbf{coocc} = \Delta\mathbf{coocc}(\mathbf{coocc})$, must be a function of only the co-occurrence matrix, **coocc**. If additional parameters are needed, their dynamics must also be represented in differential equations. As it turned out, however, the co-occurrences gradient can only be found if the structure of the graph is taken into account, i.e., **coocc** is not sufficient. Table 1 exemplifies such insufficiency by showing that a centrosymmetric **coocc** matrix (middle) does not result in a equally centrosymmetric Δ**coocc** matrix (right).

Table 1. An example of a adjacency matrix (left) whose row-pairwise *co-occurrence (coocc)* matrix (middle) is not sufficient to calculate the expected *coocc* gradient (right). For the sake of clarity, the gradient values w.r.t. the edge switching chain (right) are shown without normalization by the number of possible swaps trials per step, $|E|^2 = 11^2$.

	c_1	c_2	c_3	c_4	c_5
r_1	0	0	0	1	1
r_2	0	0	1	0	1
r_3	0	1	1	1	0
r_4	1	0	0	1	0
r_5	0	0	0	1	1

	r_1	r_2	r_3	r_4	r_5
r_1	–	1	1	1	2
r_2	1	–	1	0	1
r_3	1	1	–	1	1
r_4	1	0	1	–	1
r_5	2	1	1	1	–

	r_1	r_2	r_3	r_4	r_5
r_1	–	−4	6	0	−16
r_2	−4	–	2	12	−4
r_3	6	2	–	−2	6
r_4	0	12	−2	–	0
r_5	−16	−4	6	0	–

In fact, the expected change in co-occurrence, a node pairwise relation, can only be given by the interaction between the neighborhoods of three nodes. This becomes clear once we realize that $coocc(i, j)$ can only be changed if the neighborhood of a third node k is modified since the degree sequences are fixed. In turn, the dynamics of the node

3-wise relations can only be described by 4-wise relations, and so on. Therefore, we conclude that a full description of the dynamics of the mixing chains w.r.t. the pairwise co-occurrences is not feasible.

3 The Benchmarking Problem

In general, comparing different system implementations is a non-trivial task. The reason is that plenty of parameters influence the final system behavior, such as the underlying system architecture, the selected algorithms, the chosen software implementation language, compilers, or communication and memory infrastructures. Besides, the performance of a system can heavily depend on the input data, in particular if adaptive ("*self-tuning*") methods are used. Thus, fairly comparing implementations requires an in-depth analysis of the relevant factors and the target application domains first.

In this context, we distinguish between the *application* or *problem* (the actual task to be carried out), the employed *model* or *algorithm* and its final *implementation* on a specific *architecture*. The latter three make up the final *system solution* that we are evaluating.

Let us look more closely at an example: We define the *application* or *problem* as "recommend movies to a client who has already watched several other movies", a generic task, e.g., in a video streaming service. The choice of an appropriate *algorithm* for this task is crucial for the overall system behavior: We can, e.g., select a graph-based approach as discussed in this chapter, statistical analysis, or machine learning based methods [15]. Each of those can be implemented in pure software on a generic computer architecture, in hardware, or in a hybrid hardware/software setting that combines programmable architectures such as central processing units (CPUs) with hardware accelerators. The selection of an appropriate underlying system *architecture* is strongly linked to the chosen algorithm, since there may be strong interactions between those two. Some algorithms are more friendly for being implemented in hardware or accelerators (in particular if they allow high parallel processing), while others may fit more to programmable (i.e., in general sequential or control-driven) architectures. Thus, fixing *algorithm* and *architecture* in the system design flow is an iterative and heavily interdependent process that requires a deep understanding of both domains and the target *application*, since the latter may impose additional restrictions or constraints on the other ones. In particular, low-level parameters such as selecting appropriate data structures for efficient memory accesses or custom data types with reduced-precision can lead to strong increases in performance and energy efficiency, but may also impact the QoR (see Chapter 4).

However, evaluating/comparing *systems* always requires well-defined *metrics*. In order to allow comparisons over architectural borders, those metrics need to be independent of the underlying system architecture and/or employed software. "Operations per second" for example is still a widespread metric in the high-performance computing (HPC) domain, but cannot be applied to systems that incorporate hardware accelerators (mainly data-flow architectures with hard-wired circuits) in which no "operations" exist. Thus, we propose application-level metrics that are not related to the selected algorithms or architectures. Examples are "run-time for a specific task", "consumed energy for a run", and "achieved QoR for a specific task".

In the above-mentioned example "recommend movies to a client who has already watched several other movies", we could, e.g., compare a software implementation running on a CPU-based cluster with a (hybrid) hardware-accelerated architecture. After deciding that we are going to implement a graph-based approach over other available options, we can still select the specific algorithm for the LA part (e.g., Edge Switching (ES) or Curveball (CB), see Sect. 4), the number of processing elements (PEs), the amount of hardware acceleration (if any), the memory hierarchy, communication infrastructure, the data structure (e.g., matrix vs. adjacency list), and the data types (e.g., floating-point precision) that impact both storage demands and required computational effort. It is obvious that a large number of degrees of freedom leads to an overwhelming amount of possible *system solutions* that all solve the same task, but with different characteristics.

While "run-time for a specific task" and "consumed energy for a run" can be measured or estimated with rather straight-forward approaches, determining the "achieved QoR for a specific task" is much harder to quantify. The reason is that in general multiple ways for measuring quality exist that must be investigated more specifically in order to determine which metric provides the most meaningful insights for the specified application.

In addition, stochastic parts of the selected algorithm (e.g., based on former system states, random numbers, or early stopping criteria/heuristics) may even lead to variations over different runs with the same input data. Thus, a robust quality metric not only needs to provide a meaningful quantitative statement of the achieved QoR but should also be determined in a way that minimizes stochastic impacts on the result. In the LA part of our example ("recommendation system"), we could, e.g., use the PPV_k [24 SPP] as a direct measure of the results or autocorrelation [6 SPP] or perturbation [25] as QoR measure for the mixing itself.

A generic approach for tackling these issues are *benchmark sets* that try to cover specific application areas with typical data points. Most of them consist of so-called *batteries* that combine multiple tests into larger task lists to minimize set up/initialization and read-out overhead and to reduce stochastic effects.

In order to stop a stochastic process when a sufficient QoR is achieved, we employ *heuristics* that perform online tracking of specific QoR measures together with desired target values (so-called *early stopping criteria*). Once the target is achieved, the processing is stopped. For the Link Assessment (LA) problem with the ES chain, we have analyzed how the PPV_k changes over the number of samples and swaps throughout the processing [4 SPP]. We are using two data sets, the Netflix competition data set and a medium-size MovieLens data set[2]. More detailed insights are given in Fig. 5.

Figure 4 and Fig. 5 clearly show that the PPV_k saturates abruptly when a specific number of samples or swaps is achieved (a so-called *phase transition*). From this moment on, further processing does not increase the QoR any more. Thus, we can stop when we detect the phase transition of the PPV_k and use this as an early stopping criterion for this task.

[2] The 100k MovieLens data set, available from http://grouplens.org/datasets/movielens/.

From this criterion, we can derive an appropriate heuristic that we incorporate into the final implementation. One crucial aspect for such a heuristic is its *stability*, i.e., it must be ensured that it works reliably for the allowed range of input data sets for a given application and that it stops the processing at the earliest possible time when the desired QoR is achieved. We present appropriate heuristics for the LA in Sect. 5.1.

4 Edge Switching vs. Curveball

Generating random samples from the fixed degree sequence model (FDSM) remains the most accurate method for estimating the expected co-occurrences between nodes, and therefore also for performing the Link Assessment (LA). Exact sampling schemes, where random graph samples are generated from scratch and exactly uniformly at random, were proposed but their computational complexity is $O(n^3)$ [9]. Most commonly, the random graphs are generated by sequentially mixing the original graph's edges, specifically using the Edge Switching (ES) Markov chain. Strona et al. [25] proposed a new algorithm, coined the Curveball (CB), which instead of switching a pair of edges, randomly trades the neighborhoods of two nodes. The CB was quickly proven to converge to the uniform distribution and adapted for different types of graphs [5] (see Chapter 2 for more details about the Curveball algorithm).

The mixing time of a Markov chain refers to the number of steps in the chain required to reach any possible state with equal probability[3], i.e., to disassociate the final, random state from the initial state [21]. While the true mixing time of neither Markov chain, the ES or the CB, is known, first empirical results suggested that the CB was a more efficient method of randomizing a graph [5,25]. These are based on the perturbation score and discussed in terms of the number of steps in the respective Markov chains. In practice, however, one CB step may take much longer than one ES step, so an actual runtime comparison between implementations is more meaningful.

In this section, we show the runtime comparison between two versions of the CB algorithm and an ES implementation. The Sorted-lists Curveball (SCB) iterates through two randomly selected nodes' neighborhoods (lists) in order to find, shuffle and re-assign the disjoint set of neighbors. Although finding the disjoint set is facilitated by keeping sorted lists, after the re-assignment they must be re-sorted in preparation for the next trade, so the overall complexity is $O(deg_{max} \times \log deg_{max})$ per trade, where deg_{max} is the maximum node degree of the network. The Hashed-lists Curveball avoids sorting the lists by creating a temporary hash-map of each neighborhood. The complexity of the HCB depends on the properties of the hash-map used, but in general is between $O(deg_{max})$ and $O(deg^2_{max})$ per trade. Finally, the ES uses two redundant data structures to accomplish a complexity of $O(1)$ per swap: the adjacency lists to randomly pick two existing edges; and the adjacency matrix to check whether they can be swapped.

4.1 Perturbation Score

The perturbation score [25] is the number of different entries between the adjacency matrices of the original graph and the random sample. Since each step in the edge

[3] Within an arbitrary error margin.

switching chain can only swap two edges, at most four entries of the adjacency matrix are modified in each step. The perturbation score, although maybe not a true estimator for the total mixing time, is a direct measure of the distance of the shortest path between two states of the ES chain.

The first comparison between the randomization algorithms was conducted using the Netflix Prize dataset. Figure 1 shows the relative perturbation[4] vs. the runtime for a curveball (the SCB) and the ES implementation (shown are averages of at least 10 repetitions). The bottom-most subplot refers to the complete dataset, while the first two refer to random subsets of 10000 and 100000 users, respectively. Since the curveball algorithm is sensitive to the number of adjacency lists (=number of columns in the adjacency matrix), both Movies x Users and Users x Movies representations are simulated. In this analysis, irrespective of the amount of data and number of adjacency lists, the ES implementation is at least 2× faster than SCB.

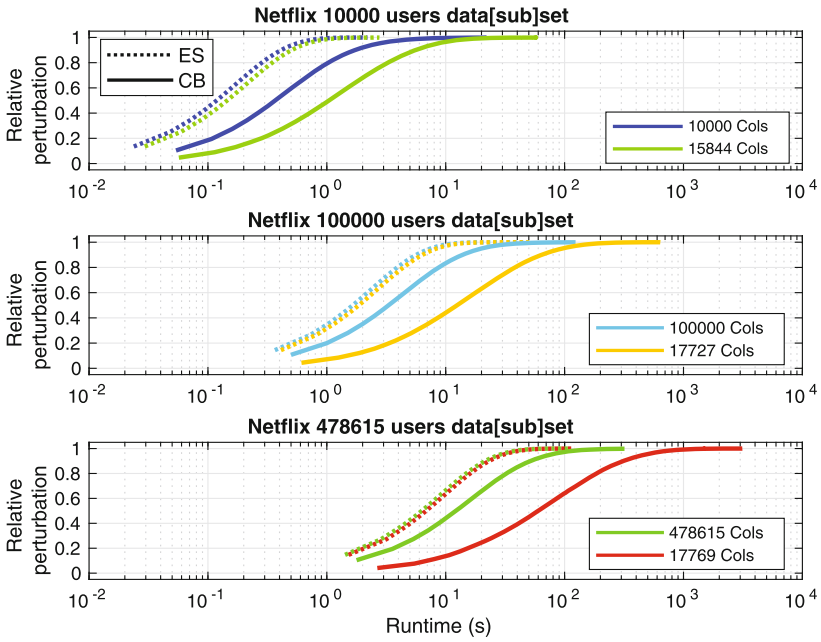

Fig. 1. Relative perturbation achieved by ES and SCB vs runtime for different subsets of the Netflix Prize dataset. Machine: Intel(R) Xeon(R) E5 2640v3 @ 2.60 GHz.

Besides the surprising results showing that the ES implementation using both the adjacency lists and matrix is faster than the CB based on sorted adjacency lists (further discussed in Sect. 4.2), another interesting effect can be seen in Fig. 1: Mixing the

[4] We define the relative perturbation as the perturbation score normalized by its maximum value among all mixing chains and repetition runs.

Users side of the bipartite network is always faster than mixing the Movies side, irrespective of the number of users and movies. An explanation for that may be in the degree distribution of the two types of nodes, users and movies, do not show the same shape. Figure 2 shows the degree distribution densities for users and movies nodes for a subset of 100000 randomly selected users of the Netflix Prize dataset. While relatively more movies have very low or very high degrees, user degrees concentrate in the middle, a trend that holds for any subset of randomly selected users (not shown). Given that a curveball trade consists of shuffling the disjoint neighborhoods of two randomly selected nodes, a higher mixing efficiency is expected when the two nodes have similar degrees. Therefore, we conclude that, when mixing bipartite graphs using the curveball Markov chain, it is advantageous to perform the trades between nodes from the least skewed degree distribution side, at least with respect to the perturbation score vs. the runtime. Note that the ratio between the number of nodes of the two sides does not play a major role w.r.t. the runtime (see Fig. 1), contrary to the belief of the original curveball algorithm inventors [25].

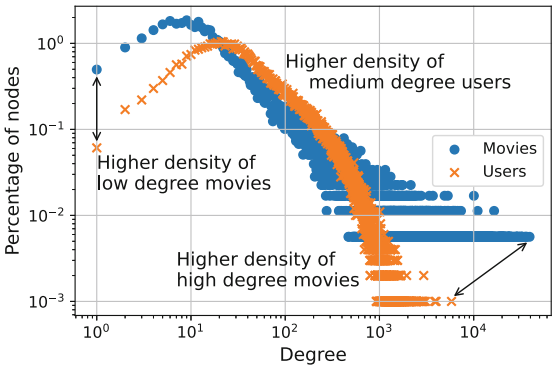

Fig. 2. The degree distribution densities of users and movies for a subset of the Netflix Prize dataset containing good ratings from 100000 random users. Users' degrees are more concentrated in the middle while movies' degrees have a more skewed distribution, with many very low and many very high degree movies.

4.2 Runtime Comparison with NetworKit

In [6 SPP], an I/O-efficient implementation of the curveball trades for simple, unipartite graphs is proposed. Its key feature is the introduction of a trade sequence that is lexicographically sorted before the curveball trades are performed, resulting in a more efficient memory access (see Chapter 2 or [6 SPP] for more details). In cooperation with the Group of Algorithm Engineering from the Goethe University Frankfurt, we compare our ES, SCB, and HCB implementations to a bipartite version of their curveball.

Figure 3 shows the runtime comparison results between our ES, SCB, and HCB and the NetworKit CB [6 SPP] implementations. Each mixing chain executes 20 super

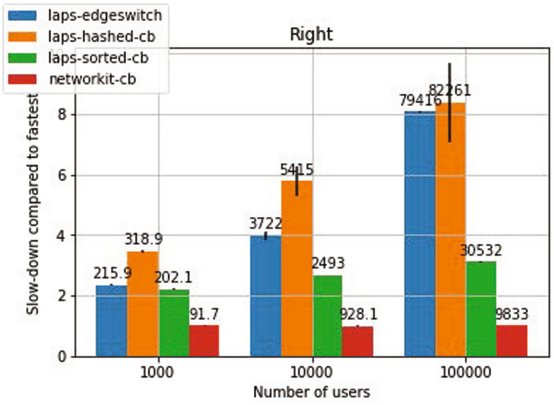

Fig. 3. Runtime comparison of the ES and three curveball implementations. The numbers over the colored bars indicate the average runtime in milliseconds for 10 super steps (see text). The black, thin bars indicate the standard deviation of 10 runs. NetworKit CB is consistently faster (at least 2x faster) and scales better than our ES, HCB, and SCB implementations. Machine: Intel(R) Xeon(R) CPU E5-2630v3 @ 2.40 GHz. (Color figure online)

steps, i.e., $10 \times \#Users$ for the curveball and $10 \times \#Edges$[5] for the edge switching, which considers, in expectation, 20 times the state of each edge, which is when both ES and CB converge in quality according to the autocorrelation thinning factor [6 SPP].

Figure 3 shows that NetworKit CB is at least $2\times$ faster than our CB implementations, as well as faster than the ES. From Fig. 1 we see that our ES is between $1.5\times$ to $2.5\times$ faster than the SCB, which in turn is between $2\times$ and $3\times$ slower than NetworKit CB. Therefore it is safe to assume that there is a CB implementation that is at least as fast as our ES even according to the perturbation[6]. Furthermore, it becomes evident from Fig. 3 that the CB–both our SCB and NetworKit's–scales consistently with the size of the graph, while the ES appears to have higher factors.

With these results, we conclude that the Curveball algorithm can indeed be efficiently implemented in software, and scales better than the ES. However, it also became clear that there can be huge differences in results interpretation depending on the quality metric being regarded (perturbation or auto-correlation), and it is not clear which is the most relevant.

5 Phase Transition and Heuristics

Instead of a steady increase in the quality of the LA with the underlying MCMC parameters, mainly the mixing length and number of samples, we actually see a flat low

[5] Notice the factor $0.5\times$ between the super step definition in [6 SPP] and here. This is due to the representation of unipartite, undirected graphs [6 SPP] requiring duplicated edges, one in each direction, while bipartite graphs do not.

[6] Unfortunately, a direct comparison w.r.t. the perturbation turned out to be difficult because of incompatibilities between the structures of the source codes.

quality followed by a sudden and steep increase. This phase transition-like behavior was first reported in [4 SPP], further discussed in [3 SPP], and is shown in Fig. 4. The LA quality, measured by the PPV_k, is plotted against the number of samples, the main parameter w.r.t. the total runtime of the method.

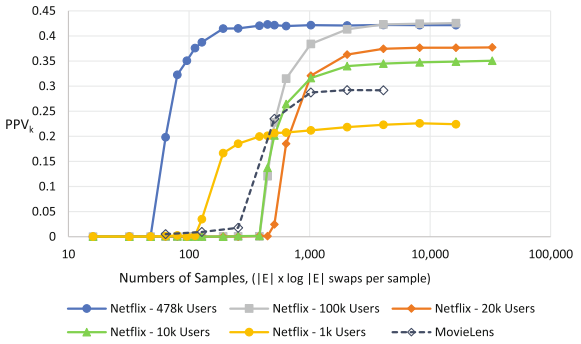

Fig. 4. Link assessment quality (PPV_V) over number of samples. For a wide variety of data sets, narrow phase transitions are present. Similar phase transitions are also seen for the number of edge swaps (see Fig. 7 in [3 SPP]). (Color figure online)

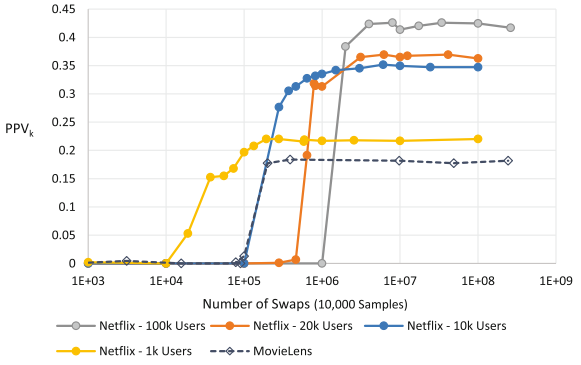

Fig. 5. Quality over number of swaps. For a wide variety of data sets, narrow phase transitions are present, similarly to what can be observed varying the number of samples. (see Fig. 2 in [4 SPP]). (Color figure online)

The complete Netflix dataset (blue), e.g., requires 384 samples to reach a PPV_k of 0.4206 ± 0.0019, while 16,384 samples only increase this value to 0.4217 ± 0.0012. While, on one hand, it indicates that a low number of samples is required, the steep transition also cautions us against taking too few samples, since 64 samples instead of 384 would result in roughly half the quality (0.20 ± 0.03) and 48 samples (0.001 ± 0.001) is no better than a random guess. Therefore, this is not mainly a trade-off problem, where more resources (samples) bring better quality, but rather a threshold problem, at which

the quality transitions steeply from its minimum to its maximum. Finding this threshold can be done via online heuristics, as is discussed in Sect. 5.1.

5.1 Heuristics for MCMC Parameters

Continuing the sampling procedure after the maximum LA quality is reached results in increased runtime for no benefit. Therefore, being able to reliably assess when maximum quality has been reached can represent great speedups for the complete link assessment method. For example, compared to the commonly used 10,000 samples, we see in Fig. 4 that, for the complete Netflix dataset, the LA reaches maximum quality at around 384 samples, which would represent a speedup of $> 25\times$.

Figure 4 also shows that the tipping point depends on the dataset. However, we were not able to find correlations between the input dataset and the required number of samples, and so an analytic formula could not be derived.

Even though the mechanism that causes the phase transitions is not yet completely understood, we know that it must be related to the stability of the ranking of the most significant pair of nodes. We have tested multiple methods to evaluate the stability of the final result by comparing it with the previous one [4 SPP, 3 SPP]. A good estimator for the final quality should also present a phase transition-like behavior, as does the quality itself. In [3 SPP] we described in detail our successive attempts to find a reliable heuristic, going from the number of matching pairs at the very top of the ranking, to more sophisticated correlations methods, and finally the internal PPV_k method. The internal PPV_k method consists of creating an internal ground truth based on the top ranked node pairs at each iteration and using it to calculate the PPV_k after a new (group of) sample(s) is drawn. This measure turned out to be stable and well correlated to the observed phase transition of the actual quality, based on the real ground truth.

A rather similar heuristic, in the sense that it also presents a phase transition-like behavior, was devised for assessing the required amount of mixing (w.r.t. the number of edge swaps) necessary [3 SPP]. It is based on the fact that the average $coocc_{FDSM}(a,b)$ of two nodes a, b only depends on their degrees. Therefore, we expect to see a converging behavior of multiple node pairs that have the same degree pair if the amount of mixing is sufficient (see Fig. 10 in [3 SPP]).

Table 2 summarizes the speedups achieved by applying each and the two heuristics for the number of swaps and samples presented in [4 SPP, 3 SPP]. In all cases, the heuristics accelerate the overall LA (all overheads are accounted for) when compared to the safe number of swaps ($|E| \times \log |E|$) and samples (10000) without any significant degradation of its quality. The largest graph seems to benefit the most from either and both heuristics. For the smallest graph, the Movielens data, the overhead of the swaps heuristic becomes significant, shadowing the increased randomization speed during the actual sampling. Also, it still requires around one-third of the 10000 samples for the internal PPV_k to converge.

Notice that we can clearly see an interdependence between the number of swaps and the number of samples. For example, using a more conservative set of parameters for the swaps heuristic results in 8×10^7 edge swaps and 454 samples for the complete Netflix data, almost ten times more swaps but 4 times fewer samples as the less conservative alternative. This is expected since the relevance (or entropy) of each new sample is

Table 2. Runtime and quality comparisons of the LA with and without heuristics [3 SPP]. The swaps heuristic has three parameters: the convergence threshold, θ_{min}; the number of groups of node pairs with the same degree pair, N_g; and the number of node pairs per group, N_p. The samples heuristic also takes three parameters: the internal PPV_k threshold, α; the size of the internal ground truth, k'; and the number of samples between evaluations, $samples_{step}$.

Data set	Samples	Edge swaps	Runtime[a]	PPV
State-of-the-art				
Netflix, 487k users	10,000	10^9	20 h	0.422
Netflix, 100k users	10,000	2.6×10^8	5.5 h	0.425
MoviesLens	10,000	1.5×10^7	877 s	0.290
Swap heuristic:				
Netflix, 478k users[c]	10,000	6.3×10^6 (**160x**)	0.2 + 6.8 h (2.9x)	0.418 (−0.9%)
Netflix, 478k users[d]	10,000	8.0×10^7 (**13x**)	0.8 + 9.6 h (1.9x)	0.424 (+0.1%)
Netflix, 100k users[b]	10,000	6.8×10^7 (**4x**)	0.2 + 3.4 h (1.5x)	0.424 (−0.2%)
MovieLens[b]	10,000	1.5×10^6 (**10x**)	24 + 554 s (1.5x)	0.290 (0.0%)
Sample heuristic[e]:				
Netflix, 487k users	640	10^9	1.42 h (**14x**)	0.418 (−0.9%)
Netflix, 100k users	2,944	2.6×10^8	1.66 h (**3.3x**)	0.419 (−1.3%)
MovieLens	3,456	1.5×10^7	326 s (**2.7x**)	0.291 (+1.0%)
Combined heuristic[e]:				
Netflix, 487k users[c]	1,664	6.3×10^6	0.2+1.3 h (**13.3x**)	0.420 (−0.5%)
Netflix, 487k users[d]	454.4	8.0×10^7	0.8+0.5 h (**15.4x**)	0.419 (−0.7%)
Netflix, 100k users[b]	3,072	6.8×10^7	0.2+1.2 h (**3.9x**)	0.420 (−1.1%)
MoviesLens[b]	3,949	1.5×10^6	24+252 s (**3.2x**)	0.290 (0.0%)

[a]When appropriate, reported as (Swap heuristic runtime) + (LA core runtime)
[b]Swap heuristic parameters: $N_p = 4$; $N_g = 24$; $\theta_{min} = 0.01$
[c]Swap heuristic parameters: $N_p = 36$; $N_g = 36$; $\theta_{min} = 0.05$
[d]Swap heuristic parameters: $N_p = 36$; $N_g = 36$; $\theta_{min} = 0.005$
[e]Sample heuristic parameters: $samples_{step} = 128$; $k' = 0.2\%\ |V_R|^2$; $\alpha = 0.95$.

directly related to the amount of mixing between samples - the more independent the samples, the more information each of them adds to the poll. In this case, the more conservative choice is slightly faster.

The trade-off between the amount of mixing between samples and the required total number of samples provides an effective measure of the quality of each random sample. As the quality measure is independent of the mixing chains, we have used it to compare the effectiveness of different chains and their runtimes. An online heuristic that optimizes this trade-off is also under investigation.

5.2 Phase Transitions as Mixing Quality Estimation

Earlier results from the implementation of heuristics to find appropriate parameters for the MCMC sampling showed that there exists a trade-off between the amount of mixing

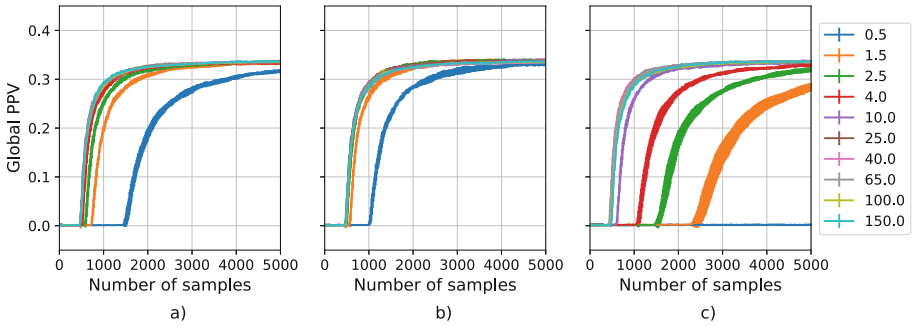

Fig. 6. The phase transitions in the link assessment quality of the Netflix 10k dataset for increasing mixing lengths of a) Edge Switching, b) Curveball in Movies, and c) Curveball in Users. The vertical line-width represent the standard deviation of 10 independent simulations. Mixing lengths are given in super steps, where 1 super step (Sect. 4.2) is a) $|E|/2$, b) $|M|/2$, and c) $|U|/2$.

and the number of samples required for convergence (Table 2). This insight led us to investigate how the PPV_k phase transitions behave when we vary the mixing length of the chains.

Figure 6 shows the LA quality phase transitions w.r.t. the number of samples for increasing mixing chain lengths and three different mixing chains. As expected, the phase transitions are shifted to the right (more samples are required) as the mixing length decreases. If the mixing length is sufficient, however, increasing it further may not significantly change the result. This behavior can be explained by the correlation (or level of independence) between consecutive samples, which is expected to decrease exponentially with the amount of mixing. If enough mixing is performed, the sampling procedure became virtually as efficient as it can be, as if each sample was drawn uniformly at random. If the correlation between each consecutive sample is measurable, the sampling efficiency is degraded, therefore the phase transitions both start later and become less steep.

Figure 6 also shows that there can be a great difference in the efficiency of the Curveball algorithm whether we choose to mix (b) movies or (c) users, even if the mixing lengths are normalized w.r.t. the adjacency matrix dimensions. Similar behavior was already observed in Fig. 1, where mixing the users' neighbors was faster in reaching the maximum perturbation. Surprisingly, however, mixing the movies side instead of users is more effective when the LA quality is regarded. To better expose this controversy, we can find the number of samples required to reach a certain PPV_k threshold, say 0.3 in this case. If the perturbation was a good predictor of the sampling efficiency, we would expect the number of samples to be the lowest only when the maximum perturbation is reached.

Figure 7 shows the perturbation and the number of samples required to reach the threshold PPV_k of 0.3 versus the amount of mixing. For the edge switching (ES) and the Curveball in users (CB_C), there seems to exist a strong (inverse) correlation between the perturbation and the number of samples. When the Curveball mixes the movies neighborhoods (CB_R), however, the perturbation is far too conservative. While the

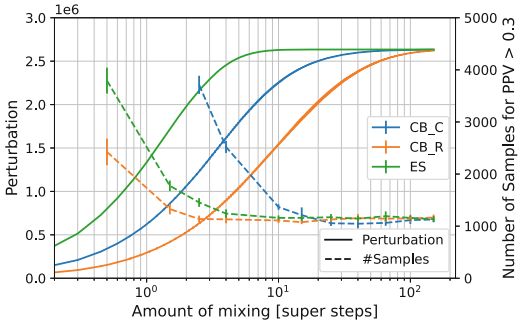

Fig. 7. The perturbation (left axis) and the number of samples until the PPV_k reaches 0.3 (right axis) vs the mixing length for the Netflix 10k dataset. The error bars represent the standard deviation of 10 independent simulations. Mixing chains: Curveball in Users (CB_C); Curveball in Movies (CB_R); Edge Switching (ES).

minimum number of samples of around 1200 is reached at 2.5 super steps, only at 150 super steps does the perturbation reaches its maximum value - an overdo of 60×.

6 Summary and Conclusion

In this chapter, we summarize strategies for increasing the sampling efficiency for the Link Assessment (LA) problem. Although we provide specific results for the latter one, our approach is generic and can be applied to related applications. Since in many cases no closed-form solutions exist, stochastic Markov chain Monte Carlo (MCMC) need to be employed. However, their main drawbacks are the high computational demand and the lack of reproducibility of exact numerical results due to their stochastic components. In addition, different algorithms may be used for generating the MCMC samples (edge switching vs. curveball), and their performance on different compute architectures can vary strongly. In many cases, it is not clear which algorithm is the best one for a specific data set on a specific architecture with respect to performance or quality.

Thus, we highlight the importance of application-level benchmark sets together with application-level, numerical measures for the quality of results (QoR) of specific runs. Such benchmarks allow a fair and quantitative comparison of non-architecturally linked metrics such as “energy per run”, “time per run”, or “quality of the results”. In addition, for many real-world data sets, we do not have any kind of ground truth that could serve as a test oracle when evaluating the achieved quality. In order to overcome this issue, we propose to construct meaningful benchmark batteries (together with ground truths) for specific application domains artificially that cover the main tasks and the corner cases equally.

Furthermore, we consider the PPV_k being a good quality measure for the LA problem. However, since it is strongly linked to the ground truth not available in many cases, we have investigated alternative metrics such as autocorrelation or perturbation. Those also allow the construction of unsupervised systems that are able to determine a “sufficient” QoR on their own. We clearly observe correlations between the latter ones and

the PPV_k in Fig. 7, but a comprehensive analysis of any formal causalities between them is currently ongoing.

Finally, we propose a working heuristic for an LA solver based on edge switching that exploits observable phase transitions of the PPV_k as an early stopping criterion for the MCMC process. This heuristic provides speedups of up to 15.4× compared to solvers that use conservative approaches.

References

1. Barabási, A.L., Albert, R., Jeong, H.: Mean-field theory for scale-free random networks. Physica A: Stat. Mech. Appl. **272**(1), 173–187 (1999). https://doi.org/10.1016/S0378-4371(99)00291-5
2. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowl. Based Syst. **46**, 109–132 (2013). https://doi.org/10.1016/j.knosys.2013.03.012

3 SPP. Brugger, C., et al.: Increasing sampling efficiency for the fixed degree sequence model with phase transitions. Soc. Netw. Anal. Min. **6**(1), 1–14 (2016). https://doi.org/10.1007/s13278-016-0407-0

4 SPP. Brugger, C., et al.: Exploiting phase transitions for the efficient sampling of the fixed degree sequence model. In: ASONAM, pp. 308–313. ACM (2015). https://doi.org/10.1145/2808797.2809388

5. Carstens, C.J., Berger, A., Strona, G.: Curveball: a new generation of sampling algorithms for graphs with fixed degree sequence. CoRR abs/1609.05137 (2016). https://doi.org/10.1016/j.mex.2018.06.018

6 SPP. Carstens, C.J., Hamann, M., Meyer, U., Penschuck, M., Tran, H., Wagner, D.: Parallel and I/O-efficient randomisation of massive networks using global curveball trades. In: ESA, pp. 11:1–11:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.ESA.2018.11

7. Duranton, M., et al.: HiPEAC vision 2019. In: European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC) (2019)
8. Fosdick, B.K., Larremore, D.B., Nishimura, J., Ugander, J.: Configuring random graph models with fixed degree sequences. SIAM Rev. **60**(2), 315–355 (2018). https://doi.org/10.1137/16M1087175
9. Genio, C.I.D., Kim, H., Toroczkai, Z., Bassler, K.E.: Efficient and exact sampling of simple graphs with given arbitrary degree sequence. CoRR abs/1002.2975 (2010)
10. Gionis, A., Mannila, H., Mielikäinen, T., Tsaparas, P.: Assessing data mining results via swap randomization. ACM Trans. Knowl. Discov. Data **1**(3), 14 (2007). https://doi.org/10.1145/1297332.1297338
11. Goh, K.I., Cusick, M.E., Valle, D., Childs, B., Vidal, M., Barabási, A.L.: The human disease network. Proc. Natl. Acad. Sci. **104**(21), 8685–8690 (2007). https://doi.org/10.1073/pnas.0701361104
12. Goldberg, D.S., Roth, F.P.: Assessing experimentally derived interactions in a small world. Proc. Natl. Acad. Sci. **100**(8), 4372–4376 (2003). https://doi.org/10.1073/pnas.0735871100
13. Gotelli, N.J.: Null model analysis of species co-occurrence patterns. Ecology **81**(9), 2606–2621 (2000). https://doi.org/10.1890/0012-9658(2000)081[2606:NMAOSC]2.0.CO;2
14. Gotelli, N.J., Ulrich, W.: The empirical Bayes approach as a tool to identify non-random species associations. Oecologia **162**(2), 463–477 (2010). https://doi.org/10.1007/s00442-009-1474-y

15. Isinkaye, F., Folajimi, Y., Ojokoh, B.: Recommendation systems: principles, methods and evaluation. Egypt. Inform. J. **16**(3), 261–273 (2015). https://doi.org/10.1016/j.eij.2015.06.005
16. Leicht, E.A., Holme, P., Newman, M.E.J.: Vertex similarity in networks. Phys. Rev. E **73**, 026120 (2006). https://doi.org/10.1103/PhysRevE.73.026120
17. Lü, L., Zhou, T.: Link prediction in complex networks: a survey. Physica A: Stat. Mech. Appl. **390**(6), 1150–1170 (2011). https://doi.org/10.1016/j.physa.2010.11.027
18. Newman, M.: Networks. OUP, Oxford (2018)
19. Newman, M.E.J.: Networks: An Introduction. Oxford University Press, Oxford (2010). https://doi.org/10.1093/ACPROF:OSO/9780199206650.001.0001
20. Rapti, A., Tsolis, D., Sioutas, S., Tsakalidis, A.K.: A survey: mining linked cultural heritage data. In: EANN Workshops, pp. 24:1–24:6. ACM (2015). https://doi.org/10.1145/2797143.2797172
21. Rechner, S., Berger, A.: Marathon: an open source software library for the analysis of Markov-chain Monte Carlo algorithms. CoRR abs/1508.04740 (2015). https://doi.org/10.1371/journal.pone.0147935
22 SPP. Schlauch, W.E., Horvát, E.Á., Zweig, K.A.: Different flavors of randomness: comparing random graph models with fixed degree sequences. Soc. Netw. Anal. Min. **5**(1), 1–14 (2015). https://doi.org/10.1007/s13278-015-0267-z
23 SPP. Schlauch, W.E., Zweig, K.A.: Influence of the null-model on motif detection. In: ASONAM, pp. 514–519. ACM (2015). https://doi.org/10.1145/2808797.2809400
24 SPP. Spitz, A., Gimmler, A., Stoeck, T., Zweig, K.A., Horvát, E.: Assessing low-intensity relationships in complex networks. PLoS ONE **11**(4), 1–17 (2016). https://doi.org/10.1371/journal.pone.0152536
25. Strona, G., Nappo, D., Boccacci, F., Fattorini, S., San-Miguel-Ayanz, J.: A fast and unbiased procedure to randomize ecological binary matrices with fixed row and column totals. Nat. Commun. **5**(1), 4114 (2014). https://doi.org/10.1038/ncomms5114
26. Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: ICCV, pp. 843–852. IEEE Computer Society (2017). https://doi.org/10.1109/ICCV.2017.97
27. Zweig, K.A., Kaufmann, M.: A systematic approach to the one-mode projection of bipartite graphs. Soc. Netw. Anal. Min. **1**(3), 187–218 (2011). https://doi.org/10.1007/s13278-011-0021-0