Unsupervised and Universal Data Reduction and Network Sparsification Methods By Minimal Algorithmic Information Loss*

Hector Zenil[1,2,3,4], Narsis A. Kiani[1,2,3,4], Antonio Rueda-Toicen[1,4,5] and Jesper Tegnér[2,3,6]

[1] Algorithmic Dynamics Lab, Centre for Molecular Medicine, Karolinska Institute, Stockholm, Sweden

[2] Unit of Computational Medicine, Department of Medicine Solna, Karolinska Institute, Stockholm, Sweden

[3] Science for Life Laboratory (SciLifeLab), Stockholm, Sweden

[4] Algorithmic Nature Group, LABORES for the Natural and Digital Sciences, Paris, France

[5] Instituto Nacional de Bioingeniera, Universidad Central de Venezuela, Caracas, Venezuela

[6] Biological and Environmental Sciences and Engineering Division, Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Kingdom of Saudi Arabia

Abstract

We introduce a family of unsupervised domain-free and (asymptotically) model-independent algorithms based on algorithmic information theory designed to minimize the loss of any (enumerable computable) property contributing to the object's algorithmic content and thus important to preserve in the process of data dimension reduction when forcing the algorithm to delete the least important features. Being independent of any particular criterion and of general purpose, they are universal in a fundamental mathematical sense. Using suboptimal approximations of efficient (polynomial) estimations we demonstrate how to preserve network properties outperforming other (leading) algorithms for network dimension reduction. Our method preserves all graph-theoretic indices measured, ranging from degree distribution, clustering coefficient, edge betweenness, and degree and eigenvector centralities. We conclude and demonstrate numerically that our unsupervised, Minimal Information Loss Sparsification (MILS) method is robust, has the potential to maximize the preservation of all recursively enumerable features in data and networks, and achieves equal to significantly better results than other data reduction and network sparsification methods.

*Corresponding author: hector.zenil@ki.se Online implementation is freely available online at `http://www.complexitycalculator.com/MILS` and source code for R and Wolfram Language at `https://github.com/algorithmicnaturelab/MILS`

Keywords: coarse-graining; renormalization; data reduction; feature selection; graph complexity; causal segmentation.

1 Introduction and Motivation

The study of large and complex datasets, or big data, organized as networks has emerged as one of the central challenges in most areas of science and technology. Cellular and molecular networks in biology is one of the prime examples. Henceforth, a number of techniques for data dimensionality reduction, especially in the context of networks, have been developed.

Data reduction consists in the transformation of numerical or alphabetical digital information into a simplified smaller representation preserving certain properties of 'interest' which are usually defined as the *most meaningful* parts. The question in the area of data reduction is how can low dimensional structures be detected in high dimensional data. The main purpose of data dimensionality reduction involves two different sides of the same coin. On the one hand is the minimization of the loss of information and, on the other hand, the maximal preservation of the most 'meaningful' features characterizing an object (i.e. feature selection). Traditionally, such meaningful features of interest are defined in terms of a user-centric, subjective criterion. For example, linear algebraic (e.g. matrix analysis) and statistical-based dimensionality reduction techniques attempt to minimize statistical information loss under certain algebraic (interpreted as signal and noise) conditions, having as consequence the maximization of the statistical mutual information between the desired information and the dimensionality-reduced output.

However, statistical approaches and classical information theory cannot preserve computable features that do not have some statistical signature no matter how important they may be at characterizing the object (thus making the choice of preserving statistical information arbitrary and fragile) [34, 38], that is, such techniques (e.g. PCA) will miss any non-linear and algorithmic regularity if it does not show a statistical property. Because the number of algorithmic features outgrows the number of statistical (the set of statistical is a proper subset of the algorithmic) PCA, just as all other computable measures for data reduction and clustering techniques, will miss fundamental properties of interest, as it is already know that it would miss non-linear embeddings that are possible to find but impossible for e.g. statistical and linear techniques to see. Non-linear methods of dimensionality reduction, where the dimensionality reducing mapping can be non-linear. For example,

topological data analysis can reduce data by minimizing its size or dimension into a non-linear surface of low algorithmic complexity, e.g. a torus, or an S-shape function.

The success of both linear and non-linear techniques can thus be simplified by looking for the shortest specification they can achieve, for linear algorithms this is usually approached by traditional statistical techniques while for non-linear, some domain specific subset of algorithms is considered (e.g. the set of all possible geometric shapes). Here, by not constraining to a domain, we take a step forward towards more *universal* techniques free of domains and particular implementation.

For example, if datapoints can be embedded in a low-dimension subspace or topological submanifold (such as a torus) an algorithmic loss minimization algorithm would approximate such shortest description of the generative mechanism of the torus.

Here we introduce a family of semi-computable algorithms that specifically target the preservation of computable properties (thus both statistical and algorithmic) and can thus be seen as a generalization of all dimension reduction procedures.

Graphs have been used as an efficient formal structure for representing data. Network science is now central to many areas, including molecular biology, serving as a framework for reconstructing and analyzing relations among biological units [3, 14, 25, 4].

The main aim of dimension reduction in a network is to approximate a network with a sparse network. There are several methods available in the literature for graph sparsification. Chew [9] used the shortest-path distance between every pair of vertices as a criterion for sparsifying a network. The concept of cut problems has been utilized for sparsification by Benczur and Karger [6]. In what is one of the latest methods, spectral similarity of graph Laplacians has been used for sparsification by [30].

For network dimensionality reduction one may choose as a criterion the preservation of graph-theoretic properties such as graph distance, clustering coefficient or degree distribution, or a finite (usually small) combination of these or other indices. But no finitely computable approach can find all possible features of interest in a dataset. For example, all those recursively enumerable features that the set of all Turing machines can characterize, all at the same time [34], which means that the observer is forced to make an arbitrary choice of features of interest (see e.g. [38]).

We will test our algorithms on non-trivial cases against state-of-the-art algorithms, including the most sophisticated non-linear (spectral) involving simple graphs where statistical regularities are even easier to conceal and

thus may easily fool weaker, linear and computable measures [38].

This approach opens a path towards evaluating the success of all other reduction techniques and for achieving optimal reduction based on minimization of algorithmic information loss (thus the non-linear generalization of all techniques) rather than only preserving statistical or domain-specific algebraic properties. While the algorithms introduced are independent of approximating method and can be implemented using Entropy or lossless compression, here we use a method based on [39]. Our results indicate that we either match the results of the currently best algorithms and, most of the time, outperform them for both local and global graph properties.

2 Preliminaries and Background

2.1 Cellular automata

We will use space-time diagrams of cellular automata to illustrate the way in which the algorithm operates.

A cellular automaton is a computer program that applies in parallel a global rule composed of local rules on a tape of cells with symbols (e.g. binary). Thoroughly studied in [40], Elementary Cellular Automata (or ECA) are one-dimensional cellular automata that take into consideration in their local rules the cell next to the centre and the centre cell.

In the case of 1-dimensional CA it is common to introduce the *radius* of the neighbourhood template, which can be written as $\langle -r, -r+1, \ldots, r-1, r\rangle$ and has length $2r+1$ cells. With a given radius r the local rule is a function $f : \mathbb{Z}_{|S|}^{|S|^{(2r+1)}} \to \mathbb{Z}_{|S|}$ with $\mathbb{Z}_{|S|}^{|S|^{(2r+1)}}$ rules.

Elementary Cellular Automata (ECA) have a radius $r = 1$ (closest neighbours), having the neighbourhood template $\langle -1, 0, 1\rangle$, meaning that the neighbourhood comprises a central cell. From this it follows that the rule space for ECA contains $2^{2^3} = 256$ rules.

Enumeration of ECA rules: It is common to follow the lexicographic ordering scheme introduced by Wolfram [40]. According to this encoding, the 256 ECA rules can be encoded by 8-bits.

A space-time diagram captures the evolution of a cellular automaton for a given initial condition and is read from the top starting from time step $t = 0$ (the initial condition) and evolves towards the bottom of the diagram (see Fig. 2).

2.2 Graph/network theory and information theory

Definition 2.1. A graph is an ordered pair $G = (V, E)$ comprising a set V of nodes or vertices and a set E of edges or links, which are 2-element subsets of V.

Definition 2.2. The spectrum of a graph is the list of Eigenvalues from the graph adjacency matrix sorted from largest to smallest.

Definition 2.3. Given a simple graph G with $n = |E|$ vertices, its Laplacian matrix $L_{n\times n}$ is defined by $L = D - A$. where D is the degree matrix and A is the adjacency matrix of the graph.

In what follows, we will use the terms nodes and vertex, and links and edges, interchangeably.

2.3 Classical Information Theory and Shannon Entropy

Central to information theory is the concept of Shannon's information Entropy, which quantifies the average number of bits needed to store or communicate the statistical description of an object.

For an ensemble $X(R, p(x_i))$, where R is the set of possible outcomes (the random variable), $n = |R|$ and $p(x_i)$ is the probability of an outcome in R. The Shannon Entropy of X is then given by

Definition 2.4.

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{1}$$

Which implies that to calculate $H(X)$ one has to know or assume the mass distribution probability of ensemble X.

2.4 Graph complexity

The algorithmic information content of a graph denoted by $C(G)$ is given by the so-called algorithmic Coding theorem [19, 12, 28]:

$$C(G) = -\log_2 AP(G)$$

where $AP(G)$ is the Algorithmic Probability of the adjacency matrix of G defined by the output frequency probability of being produced by a random 2-dimensional Turing machine (a typical deterministic Turing machine

whose single head can also move up and down as well as left and right) starting from an empty 2-dimensional grid (instead of the typical 1-dimensional tape Turing machine) as defined in [32].

A Turing machine is a general abstraction of a computer program similar to cellular automata but sequential that given an input produces an output and halts. The Turing machine is thus an algorithmic mechanistic causal explanation of the output and is at the centre of the algorithms here introduced. The idea is to find a short Turing machine that explains an object (e.g. a network) by explaining smaller overlapping segments of the object [33, 36].

2.5 Element information value/contribution

All methods are based on the *information difference* among the elements of an object, in other words, on the *information contribution* of the elements of a system to the whole, e.g. a network. This is based on a concept of algorithmic/causal perturbation analysis as introduced in [32, 36, 37]. The procedure consists in the perturbation of all elements of a system by the removal of elements whose effects on its algorithmic information content are measured and ranked accordingly.

Technically, let G be a network with edges $e_1, \ldots, e_n \in E$ (the same can be done for nodes), $G \backslash e_i$ be G with edge e_i removed, and I the *information difference* or *information value/contribution* of e_i to G given by

$$I(G, e_i) = C(G) - C(G \backslash e_i)$$

Where $C(G)$ is the algorithmic information content of the graph G as defined in [32] (see Methods).

When taking the difference $C(G) - C(G \backslash e_i)$ by itself we will refer to it as the graph (dis)similarity between graph G and $G \backslash e_i$. I applied to graphs suggests a similarity distance between graphs based on algorithmic information content (in [35]. We show that this similarity measure can classify networks by the family they belong to, differentiating variant synthetic and natural network topologies similar to graph motifs, as shown in [24]).

In the description of the algorithm that follows, replacing the underlying methods to approximate the (algorithmic) information content by, e.g., Shannon entropy or lossless compression algorithms represents special cases of the more general algorithm based on algorithmic complexity, and thus it covers all these less powerful cases. The idea of a dynamic study/calculus of the (possible) changes that can be wrought upon an object to evaluate

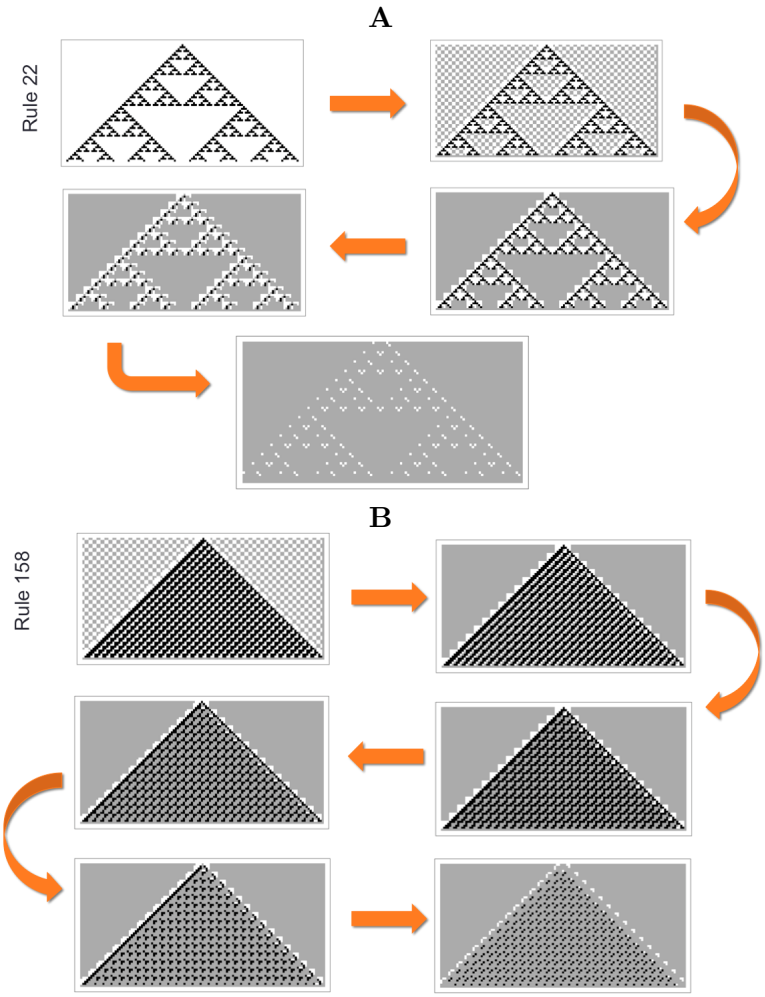

Figure 1: A: To illustrate the algorithm here is a simple optimal reduction and coarse-graining of space-time evolutions of Elementary Cellular Automata (ECA rules 22 and 158) by minimization of algorithmic information loss. Depicted are steps after application of MILS starting from original (A) and second step (B), highlighting the regions that are earmarked to be omitted (coloured in grey) versus the features that are kept and carried out along the way, thereby optimally preserving the main properties of these objects, properties whose persistence enables a ranking of such features. Unlike statistical approaches, the algorithm can also approximate (and thus preserve/extract) features that are of an algorithmic nature and which are not statistically apparent as it was in this case (see [39, 38]) and next examples.

the contribution of each of its components for different purposes was introduced in [37], and here we extend these ideas to the area of data/network dimension reduction.

2.6 Graph information content

The concept of algorithmic probability (also known as Levin's semi-measure) yields a method for approximating Kolmogorov complexity related to the frequency of patterns in the adjacency matrix of a network, including therefore the number of subgraphs in a network. The algorithmic probability [29, 19, 8] of a subgraph $H \in G$ is a measure that describes the probability that a random computer program p will produce H when run on a 2-dimensional tape universal (prefix-free[1]) Turing machine U. That is, $m(G) = \sum_{p:U(p)=H\in G} 1/2^{|p|}$. An example of a popular 2-dimensional tape Turing machine is Langton's ant [18], commonly referred to as a *Turmite*.

The probability semi-measure $m(G)$ is related to Kolmogorov complexity $C(G)$ in that $m(G)$ is at least the maximum term in the summation of programs $m(G) \geq 2^{-C(G)}$, given that the shortest program carries the greatest weight in the sum. The algorithmic Coding Theorem [12] further establishes the connection between $m(G)$ and $C(G)$ as ([19]): $|-\log_2 m(G) - C(G)| < c$ (Eq. 1), where c is some fixed constant, independent of s. The theorem implies that [12] one can estimate the Kolmogorov complexity of a graph from the frequency of production from running random programs by simply rewriting Eq. (1) as: $C(G') = -\log_2 m(G) + O(1)$.

In [13] a technique was advanced for approximating $m(G)$ (hence K) by means of a function that considers all Turing machines of increasing size (by number of states). Indeed, for small values of n states and k colours (usually 2 colours only), $D(n, k)$ is computable for values of the Busy Beaver problem [26] that are known, providing a means to numerically approximate the Kolmogorov complexity of small graphs, such as network motifs. The Coding theorem then establishes that graphs produced with lower frequency by random computer programs have higher Kolmogorov complexity, and vice versa. Here we will use the *Block decomposition method* (BDM) as an estimator of algorithmic complexity, but the algorithm and methods introduced are independent of the particular method used to approximate algorithmic complexity.

The BDM consists in decomposing the adjacency matrix of a graph into subgraphs of sizes for which complexity values have been estimated, then

[1]The group of valid programs forms a prefix-free set (no element is a prefix of any other, a property necessary to keep $0 < m(G) < 1$).

reconstructing an approximation of the Kolmogorov complexity of the graph by adding the complexity of the individual pieces according to the rules of information theory, as follows:

$$C(G) = \sum_{(r_u,n_u)\in Adj(G)_{d\times d}} \log_2(n_u) + C(r_u) \tag{2}$$

where $Adj(G)_{d\times d}$ represents the set with elements (r_u, n_u), obtained when decomposing the adjacency matrix of G into all subgraphs contained in G of size d. In each (r_u, n_u) pair, r_u is one such submatrix of the adjacency matrix and n_u its multiplicity (number of occurrences). As can be seen from the formula, repeated subgraphs only contribute to the complexity value with the subgraph BDM complexity value once plus a logarithmic term as a function of the number of occurrences. This is because the information content of subgraphs is only sub-additive, as one would expect from the growth of their description lengths. Applications of $m(G)$ and $C(G)$ have been explored in [13, 28, 27, 33], and include applications to graph theory and complex networks [32] and [33] where the technique was first introduced.

The only parameters used in the application of BDM the use of strings up to 12 bits for strings and 4 bits for arrays given the current best CTM approximations [28] and the suggestions in [39] based on an empirical distribution based on all Turing machines with up to 5 states, and no string/array overlapping in the decomposition for maximum efficiency (as it runs in linear time) and for which the error (due to boundary conditions) has been shown to be bounded [39].

2.7 Minimal Information Loss Sparsification (MILS) algorithm

MILS is an unsupervised and mostly parameter-free algorithm i.e. asymptotically not dependent of model or domain-specific as it does not need to be instructed or designed to preserve any particular property, and maximizes the preservation of all computable elements that contribute to the algorithmic information content of the data.

The MILS algorithm pseudo-code is as follows. Let G be a graph, then:

1. Calculate $G\backslash E^*_j$ for all subsets $j \in E^*$ where E^* denotes all non-empty proper subsets of edges $\{e_1, \ldots, e_n\}$.
2. Remove the edge subset E^*_j such that $C(G\backslash E^*_j) < |C(G\backslash E^*_i)|$ for all subsets i, where $|C|$ is the absolute value of C.

3. Repeat 1 such that $G := G\backslash\{e\}^*_j$ until final target size is reached.

The algorithm's time complexity class is clearly in $O(exp)$ because of the subsets' operation. A more efficient but suboptimal version of MILS iterates over single elements (nodes or edges) or singletons.

1. Calculate $G\backslash e_i$ for all $i \in \{e_1, \ldots, e_n\}$.
2. Remove edge e_j such that $C(G\backslash e_j) < |C(G\backslash e_i)|$.
3. Repeat 1 with $G := G\backslash e_j$ until final target size is reached.

We call e_j a *neutral information edge* because it is the edge that contributes less to the information content of G (in particular, it minimizes information loss or the introduction of spurious information) to the network according to the information difference when removed from the original network.

The above pseudo-code assumes that there is a unique such e_j which may not necessarily be the case. In the Supplementary Information there is a description of the exact algorithm that guarantees the uniqueness of G^n and also determines its polynomial time computational class $\leq O(n^2)$. We use this more efficient version in all our experiments reported in the Results section, and even in this limited form the procedure excels at preserving important characteristics of the networks.

MILS is, by design, *optimal* and *universal* in the computability and algorithmic-information theory sense, and only dependent on the method for approximating algorithmic complexity in the preservation of any possible feature of interest that contributes to the (algorithmic) information content of a network G such as, evidently, its degree distribution and other graph-theoretic, algebraic or topological features, even those not necessarily captured by any graph theoretic measure or classical information approach [38].

2.8 Algorithmic Information Rank

MILS immediately suggests a powerful method for ranking nodes and edges by their contribution to the information content of a network. By running MILS one produces a reversed ranking from least informative to greatest informative edges.

The pseudo-code of the algorithmic information rank (InfoRank) method of an object such as a network G can be described as follows:

1. Apply MILS on a network G.

2. Let $s = \{e_1, \ldots, e_n\}$ be the list of edges of G sorted by MILS deletion order after n iterations
3. The list of edges sorted by their information content from lowest to greatest contribution to G after reindexation is the (algorithmic) information rank (InfoRank) of the $E \in G$ (the same algorithm may be applied to nodes, or to any data element of a dataset, e.g. a row or column in a spreadsheet).

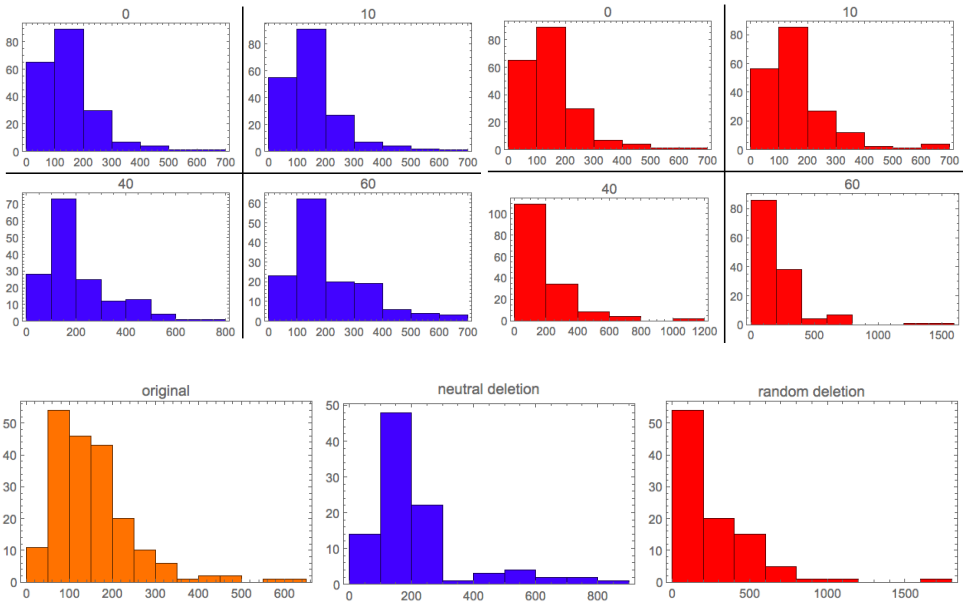

Figure 2: MILS or *neutral edge* deletion (blue) outperforms random edge deletion (red) at preserving both edge degree distribution (top, showing removed edges) and edge betweenness distribution (bottom) on an Erdös-Rényi random graph of node size 100 and low edge density ($\sim 4\%$) after up to 60 edges were removed (degree distribution comparison) and 150 edges were removed (edge betweenness) out of a total of 200 edges (notice also the scale differences on the x-axis).

The InfoRank algorithm induced by the exponential version of MILS is the most powerful but also, evidently, the most computationally expensive. The pseudo-code of the InfoRank algorithm above, however, is in the same complexity time class as the singleton version of MILS (i.e. removing a

single element at a time, or several when they collide), that is, in $O(n^2)$ time, as we will prove.

3 Results

Fig. 1 demonstrates the basic concept behind the algorithms introduced here using as a case study a simple string. Identified inflection points are natural breaking points for algorithmic partitioning, and the identification of different parts of the sequence and their contribution to the full-length sequence illustrates the way in which segments with high or low algorithmic content can be ranked, selected or preserved for dimensional reduction purposes. The method approximates the generating mechanism of each segment by finding candidate generating mechanisms that explain part of the data by finding the computer programs that produce the data and are in turn placed together in sequence to produce the full object with the sequence itself a model of the entire dataset. In other words, an algorithmic (causal) likelihood is approximated with hundreds to millions of small computer programs as a function of the algorithmic randomness of the data (the less algorithmically complex, the greater the number of candidate models) [13, 28, 39].

We demonstrate that MILS preserves essential local and global properties of synthetic and natural networks of different types and topologies, performing at least as well as but usually better than other algorithms. We took a sample of well-known and previously thoroughly studied networks from [24]. These included genetic regulatory networks, protein, power grid and social networks. We applied MILS to each of these networks and compared with two powerful sparsification methods: *Transitive reduction* [1] and *Spectral sparsification* [30]. A transitive reduction of a directed graph is therefore a graph with as few edges as possible that has the same reachability relation as the given graph. A good introduction to *spectral graph sparsification* may be found in [5]. The method was designed to reduce the network dimension based upon spectral similarity of graph Laplacians which guarantees the preservation of important properties of the graph by way of its adjacency matrix Laplacian spectrum.

Fig. 2 shows how MILS preserves the degree distribution and the edge betweenness distribution of a typical synthetically (recursively) generated Erdös-Rényi (ER) random graph (in this example of low edge density it is very sparse) compared with random edge deletion and spectral sparsification. While MILS is not significantly better at preserving the clustering coefficient of random networks, Fig. 3 shows that MILS does significantly

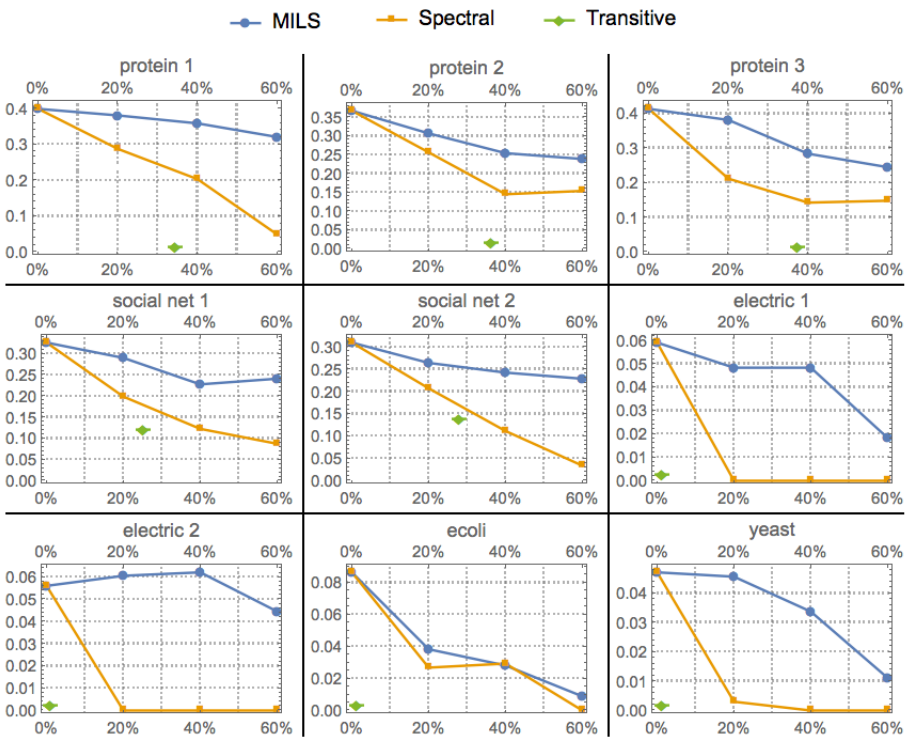

Figure 3: MILS mean clustering coefficient preservation against two other sophisticated graph sparsification methods based on graph spectral and transitive reduction techniques on biological, electric and social networks taken from [24]. The transitive method does not allow selection of edges to be deleted and in some cases it either fails to significantly reduce the network size if no cycles are present (such as, generally, in electric and genetic networks) and/or takes the clustering coefficient to 0 (e.g. for protein networks) if cycles are only local. Comparisons with other methods are unnecessary because they destroy local or global properties by design, such as clustering coefficients for the spanning tree algorithm.

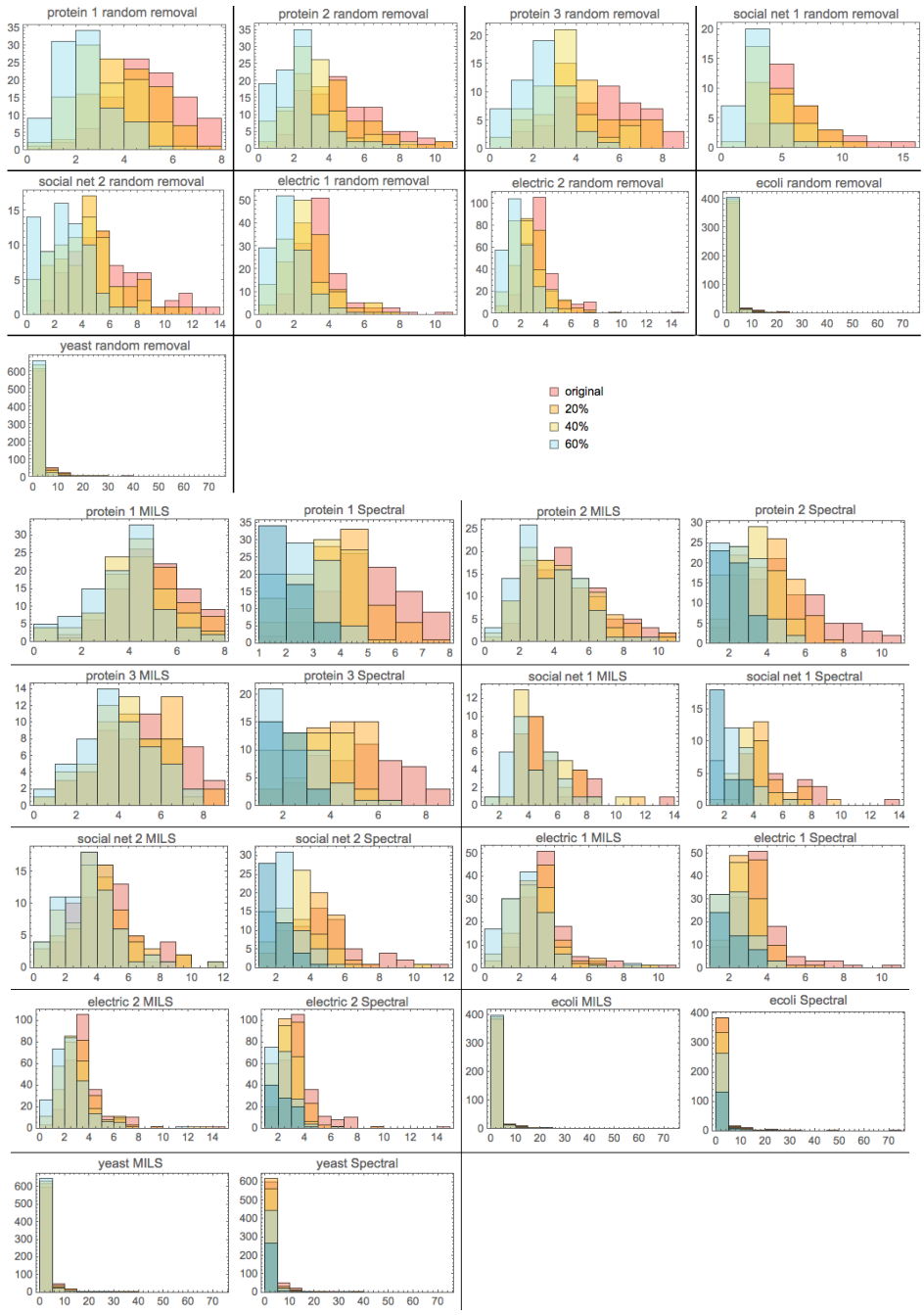

Figure 4: Histograms showing preservation of degree distribution from 20% to 80% edge removal. Green highlights the overlapping and preservation area of the distributions after random deletion (top), MILS and spectral removal (bottom pairs).

better at preserving the clustering coefficient of real-world (biological, social and electric grid) networks taken from [24], outperforming both Transitive and Spectral reduction/sparsification methods. Figs. 5(SI), 6(SI) and 7(SI) illustrate how MILS outperforms spectral sparsification at preserving edge betweenness, and degree and eigenvector centralities.

3.1 Uniqueness and time complexity

Let N be a network and $I(N,n)$ the information difference (or information value) of an element $n \in N$ to N defined by $I(N,n) = C(N) - C(N\backslash n)$. If n is a neutral element then it does not contribute to the algorithmic information content of N (by definition) thus $I(N,n) = \log n$ meaning that n is part of the dynamical causal path of N and N can regenerate n by its normal dynamical course. In general, however, if there is no element n, such that $I(N,n) = \log n$ the most neutral element n in N to be removed in the application of MILS is $I(N,n) = \min\{|C(N) - C(N\backslash n)| \sim \log(n))\}$, that is, the elements closest to $\log(n)$.

For MILS to be a proper algorithm, we need the element (e.g. node or edge) deletion procedure to determine a unique graph in a deterministic fashion. But if two or more than two elements have the same information value the algorithm becomes ambiguous.

The problem is that there may be elements such that $I(n_1) = I(n_2)$ and the algorithm cannot uniquely decide to remove n_1 or n_2 first potentially (and likely) leading to (slightly) different final results. The following algorithm tweak avoids this problem and shows that the algorithm is robust.

Theorem 3.1. *MILS is a well-defined and produces the same output for the same input (is deterministic).*

Proof sketch. Let $\{n\} \subset N$ be a subset of element of N where N can be a graph, a cellular automaton or any other object whose size is to be reduced by application of MILS. Let $I(N,n_i) = I(N,n_j)$ for element n_i and $n_j \in \{n\}$, then MILS will remove $\{n\}$ from N at the same time. This set deletion condition makes MILS a proper algorithm. □

This also produces a speedup with the MILS time complexity now ranging between the original $O(n(n-1)/2) \sim O(n^2)$ and constant time for $\{n\} = N$ i.e. when every node has the same information value and thus they are all deleted at the same time. For example, any attempt to reduce the dimension of the complete graph (either by e.g. single-node or single-edge deletion) will produce an empty graph. A minor, and perhaps useful

variation of the algorithm, is an heuristic allowing a random selection of an element when they have the same information value.

4 Discussion

Because approximations to algorithmic complexity are *lower semi-computable* (computable from below), the algorithms introduced here inherit the same limitations and advantages. One limitation is that exact values are not finitely attainable in general, but among the advantages is the universality and asymptotic robustness of the approximating methods [13, 28]. The method followed here, called the BDM (standing for Block Decomposition Method) has been designed to produce results in linear time for a non-computable task (by exchanging computation time for memory, running the algorithm and reusing the results), and its accuracy can always be increased by increasing the computational effort (only once), converging at the limit to the actual values of the algorithmic complexity of an object, independent of the chosen representation language or reference programming language (per the so-called *invariance theorem* [23]). The algorithms are designed to maximize the preservation of information while deleting the information that contributes the least to the algorithmic content, thus allowing an unbiased non-usercentric approach to algorithmic dimension reduction, that is, the reduction of the model explaining the data to a model explaining most/or the main features of the data.

The algorithm and methods introduced here are independent of the method used to approximate algorithmic complexity. Here we used a state-of-the-art method based on Algorithmic Probability as introduced previously [33, 13, 28, 39]. Our rationale is that checksum procedures and embedded decompression instructions popular in lossless compression algorithms such as LZW are not sufficiently sensitive to detect such minor changes [**?**] required in the kind of resolution needed for MILS to work. Furthermore, Shannon entropy has even greater limitations, as it is only constrained to detect trivial statistical regularities when no other updating procedure is available to properly calculate the likelihood and prior of the underlying ensemble [39].

5 Conclusion

We have demonstrated that MILS outperforms general and leading dimensionality reduction algorithms for networks and, interestingly, MILS can be

generalized to any data, as we have shown on space-time diagrams of discrete dynamical systems (cellular automata) that can also be seen as images, thus making the algorithms here introduced also applicable to challenges of image segmentation, the chief advantage being that MILS is optimal whenever using optimal methods to approximate algorithmic complexity. To test the algorithm we used a number of well-known networks commonly used in the literature to test other algorithms, on which we also applied algorithms that have been reported to lead other algorithms. The results provide evidence that MILS seems to outperform these algorithms on all indices at preserving all features of possible interest that we define as all possible features that are recursively enumerable and therefore possible to characterize using a universal Turing machine, unlike measures that are computable and cannot, even in principle, achieve such a goal. Our results are in this sense what is to be expected from theory of algorithmic information, but moreover, they also importantly demonstrates that our numerical approximations to uncomputable measures are sufficiently accurate to outperform current heuristic techniques for dimensionality reduction.

Acknowledgements

H.Z. was supported by the Swedish Research Council (Vetenskapsrådet) grant No. 2015-05299.

References

[1] A.V. Aho, M.R. Garey, J.D.Ullman, The transitive reduction of a directed graph, *SIAM Journal on Computing* 1 (2): 131–137, 1972.

[2] R. Albert, H. Jeong, A.L. Barabasi, Error and attack tolerance of complex networks, *Nature* Jul 27;406(6794):378–82, 2000.

[3] R. Albert and A.-L. Barabási, Statistical mechanics of complex networks, *Rev. Mod. Phys. 74,* 47, 2002.

[4] U. Alon, Collection of Complex Networks. Uri Alon Homepage 2007 (accessed on July 2013). http:// www.weizmann.ac.il/mcb/UriAlon/groupNetworksData.html.

[5] J. Batson, D.A. Spielman, N. Srivastava, and S.-H. Teng, Spectral Sparsification of Graphs: Theory and Algorithms, vol. 56:8, *Communications of the ACM*, 2013.

[6] A. Benczur and David R. Karger. Approximating s-t minimum cuts in O(n2)time. *In Proceedings of The Twenty-Eighth Annual ACM Symposium On The Theory Of Computing (STOC 96)*, pages 4755, May 1996

[7] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, E. Shir, A model of Internet topology using k-shell decomposition, *Proc Natl Acad Sci USA* 104: 11150–11154, 2007.

[8] G.J. Chaitin. On the length of programs for computing finite binary sequences *Journal of the ACM*, 13(4):547–569, 1966.

[9] P. Chew. There are planar graphs almost as good as the complete graph, *J. Comput. Syst. Sci.,* 39:205–219, 1989.

[10] R.L. Cilibrasi, P.M.B. Vitányi, Clustering by compression, *IEEE Trans. Inform. Theory*, 51:12, 15231545, 2005.

[11] R.L. Cilibrasi, P.M.B. Vitányi, The Google Similarity Distance, *IEEE Trans. Knowledge and Data Engineering*, 19:3, 370-383, 2007.

[12] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, 2nd Edition, Wiley-Blackwell, 2009.

[13] J.-P. Delahaye and H. Zenil, Numerical Evaluation of the Complexity of Short Strings: A Glance Into the Innermost Structure of Algorithmic Randomness, *Applied Mathematics and Computation* 219, 63–77, 2012.

[14] S.N. Dorogovtsev and J.F.F. Mendes, Evolution of Networks, *Adv. Phys.* 51, 1079, 2002.

[15] R.L.. Graham, P Hell, On the history of the minimum spanning tree problem–Annals of the History of Computing, vol 1:7, 43–57, 1985.

[16] M. Kitsak, L.K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H.E. Stanley, H.A. Makse, Identification of influential spreaders in complex networks. *Nat Phys* 6: 888–893, 2010.

[17] A.N. Kolmogorov. Three approaches to the quantitative definition of information, *Problems of Information and Transmission*, 1(1):1–7, 1965.

[18] C.G. Langton, Studying artificial life with cellular automata, *Physica D: Nonlinear Phenomena,* 22 (1-3): 120–149, 1986.

[19] L.A. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory, *Problems of Information Transmission*, 10(3):206–210, 1974.

[20] M. Li and P.M.B. Vitányi, An *Introduction to Kolmogorov Complexity and Its Applications*, 3rd. Ed. Springer, 2008.

[21] M. Li, X. Chen, X. Li, B. Ma, P.M.B. Vitányi, The similarity metric, *IEEE Trans. Inform. Th.*, 50:12, 32503264, 2004.

[22] M. Liu, B. Liu, F. Wei, Graphs determined by their (signless) Laplacian spectra, *Electronic Journal of Linear Algebra,* 22, pp. 112–124, 2011.

[23] M. Li, P.M.B. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer; 3rd ed. 2008.

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

[25] M.E.J. Newman, The structure and function of complex networks, *SIAM Review* 45 (2): 167–256, 2003.

[26] T. Rado, On Non-Computable Functions, *Bell System Technical J.*, 41, 877–884, May 1962.

[27] F. Soler-Toscano, H. Zenil, J.-P. Delahaye and N. Gauvrit, *Correspondence and Independence of Numerical Evaluations of Algorithmic Information Measures*, Computability, vol. 2, no. 2, pp. 125–140, 2013.

[28] F. Soler-Toscano, H. Zenil, J.-P. Delahaye and N. Gauvrit, *Calculating Kolmogorov Complexity from the Frequency Output Distributions of Small Turing Machines*, PLoS ONE 9(5), e96223, 2014.

[29] R.J. Solomonoff, A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 7:1–22 and 224–254, 1964.

[30] D.A. Spielman, N. Srivastava, Graph sparsification by effective resistances, *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC '08),* 563–568, 2008.

[31] D.A. Spielman, S.-H.Teng, Spectral Sparsification of Graphs, *SIAM J. Comput.,* 40(4), 981–1025, 2011.

[32] H. Zenil, F. Soler-Toscano, K. Dingle and A. Louis, Graph Automorphisms and Topological Characterization of Complex Networks by Algorithmic Information Content, Physica A: Statistical Mechanics and its Applications, vol. 404, pp. 341–358, 2014.

[33] H. Zenil, F. Soler-Toscano, J.-P. Delahaye and N. Gauvrit, *Two-Dimensional Kolmogorov Complexity and Validation of the Coding Theorem Method by Compressibility*, 2013.

[34] H. Zenil, Algorithmic Data Analytics, Small Data Matters and Correlation versus Causation. In M. Ott, W. Pietsch, J. Wernecke (eds.), *Berechenbarkeit der Welt? Philosophie und Wissenschaft im Zeitalter von Big Data*, Springer Verlag, pp. 453–475, 2017.

[35] H. Zenil, N.A. Kiani and J. Tegnér, Quantifying Loss of Information in Network-based Dimensionality Reduction Techniques, arXiv:1504.06249 , 2015.

[36] H. Zenil, N.A. Kiani and J. Tegnér, Methods of Information Theory and Algorithmic Complexity for Network Biology, arXiv:1401.3604, 2014.

[37] H Zenil, N.A. Kiani, F. Marabita, Y. Deng, S. Elias, A. Schmidt, G. Ball, J. Tegnér, An Algorithmic Information Calculus for Causal Discovery and Reprogramming Systems, https://doi.org/10.1101/185637 (preprint).

[38] H. Zenil, N.A. Kiani and J. Tegnér, Low Algorithmic Complexity Entropy-deceiving Graphs, *Physics Reviews E.* 96, 012308, 2017.

[39] H. Zenil, F. Soler-Toscano, N.A. Kiani, S. Hernández-Orozco, A. Rueda-Toicen, A Decomposition Method for Global Evaluation of Shannon Entropy and Local Estimations of Algorithmic Complexity, arXiv:1609.00110 [cs.IT].

[40] S. Wolfram, *A New Kind of Science*, Wolfram Media, Champaign IL., 2002.

Supplementary Information

5.1 Spectral sparsification

The goal of network sparsification in general is to approximate a given graph G by a sparse graph H on the same set of vertices. If H is close to G in some appropriate metric, then H can be used as a signature preserving important properties of G for faster computation after reducing the size of G and without introducing too much error. Obvious trivial sparsification methods include edge deletion by some criterion such as the outermost ones (called the k-shell method [7, 16], often used to identify the core and the periphery of the network), but most of them (such as this shell one) are rather arbitrary or ad-hoc, rather than general methods aimed at preserving important algebraic, topological or dynamical properties of the original graph, all of which constitute and contribute to the information content of the graph, that is, the necessary information to fully describe a network and reconstruct the network from that description.

A popular sparsification algorithm is the *spanning tree* [15] designed to preserve node distance but clearly destroy all other local node properties, such as the clustering coefficient. Not many non-trivial methods for network sparsification exist today. Some clearly destroy local properties, such as the spanning tree algorithm, which destroys the clustering coefficient. It is acknowledged [30, 31, 5], however, that spectral graph sparsification is among the most efficient, both at preserving important algebraic and dynamical properties of a network and in terms of fast calculation. In part the dearth of methods is due to a lack of assessment tools to decide whether one method is better than another in general terms rather than designed to preserve one or another specific graph theoretic property (e.g. the transitive edge deletion method destroys the clustering coefficient of the original graph [1]). The spectral method considered in this paper is a high-quality algorithm described in [5, 30].

Transitive reduction was introduced in [1]. A graph G is said to be *transitive* if, for every pair of vertices u and v, not necessarily distinct, $(u, v) \in G$ whenever there is a directed path in G from u to v. That is, if there is a path from a vertex x to a vertex y in graph G, there must also be a path from x to y in the transitive reduction of G, and vice versa. If a given graph is a finite directed acyclic graph, its transitive reduction is unique, and is a subgraph of the given graph.

Graph sparsification is the approximation of an arbitrary graph by a sparse graph. Here we compare MILS against random, simple (e.g. span-

ning tree) and two powerful graph sparsification and reduction methods (spectral and transitive). Spectral graph sparsification is based on the spectral similarity of graph Laplacians. A spectral sparsifier is a subgraph of the original whose Laplacian quadratic form is approximately the same as that of the original graph on all real vector inputs. Spectral graph sparsification is a stronger notion than cut sparsifiers [31] and is considered one of the most, if not the most, sophisticated sparsification or network reduction method, as it is believed to preserve some of the most important algebraic, topological and potentially dynamical properties of a network.

5.2 Graph-theoretic measures

The global clustering coefficient of G is the fraction of paths of length 2 in g that are closed over all paths of length two in G. The mean or average clustering coefficient is the mean over all local clustering coefficients of vertices of G.

The betweenness centrality for a vertex i in a connected graph is given by $\sum_{s,t \in v \wedge s \neq i \wedge t \neq i} \frac{n^i_{s,t}}{n_{s,t}}$, where $n_{s,t}$ is the number of shortest paths from s to t and $n^i_{s,t}$ is the number of shortest paths from s to t passing through i. The ratio $\frac{n^i_{s,t}}{n_{s,t}}$ is taken to be zero when there is no path from s to t.

Degree centrality is a measure of the centrality of a node in a network and is defined as the number of edges (including self-loops) that lead into or out of the node. The degree centrality of G is the list of nonnegative integers ("degree centralities") lying between 0 and $n-1$ inclusive, where n is the number of vertices of G, and identifies nodes in the network by their influence on other nodes in their immediate neighbourhood.

Eigenvector centrality is a list of normalized nonnegative numbers ("eigenvector centralities", also known as *Gould indices*) that are particular centrality measures of the vertices of a graph. Eigenvector centrality is a measure of the centrality of a node in a network based on the weighted sum of centralities of its neighbours. It therefore identifies nodes in the network that are connected to many other well-connected nodes. For undirected graphs, the vector of eigenvector centralities c satisfies the equation $c = 1/\lambda_1 a.c$, where λ_1 is the largest eigenvalue of the graph's adjacency matrix a.

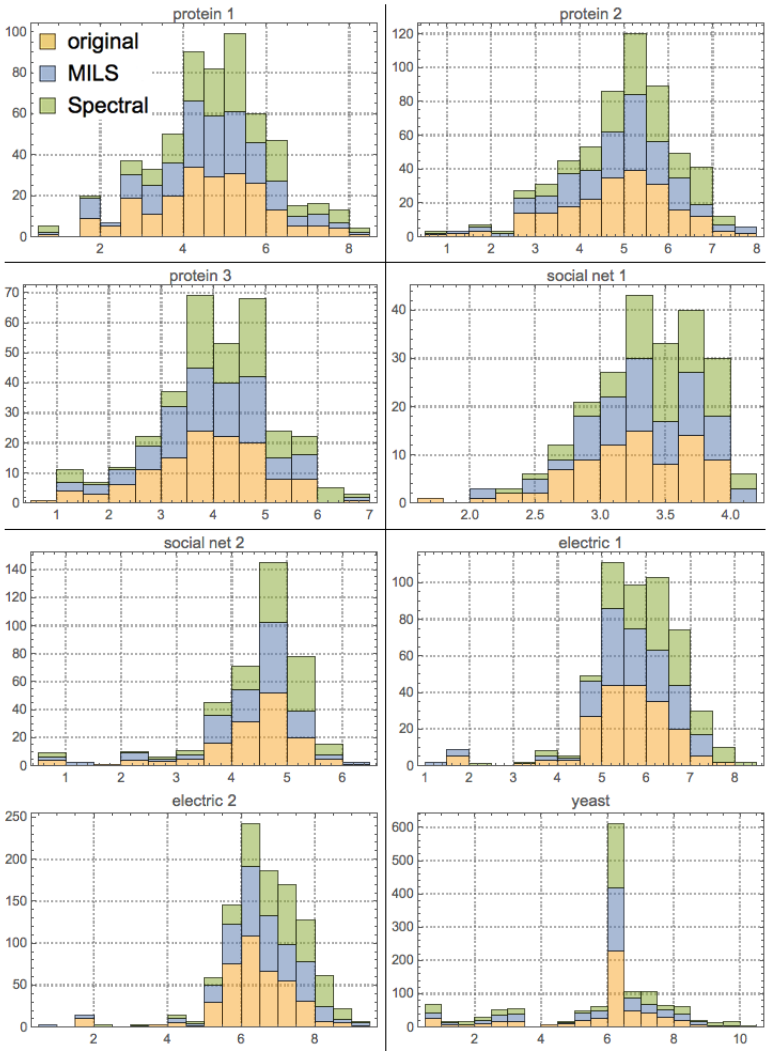

Figure 5: Stacked histograms showing edge betweenness preservation of MILS versus spectral sparsification across different families of networks. The similarity in height of each segment is an indication of the preservation of such properties. Blue bars (MILS) approximate yellow (original) bars better than spectral sparsification. On average MILS was 1.5 times the edge betweenness distribution of these representative graphs measured by the area similarity of the respective bars.

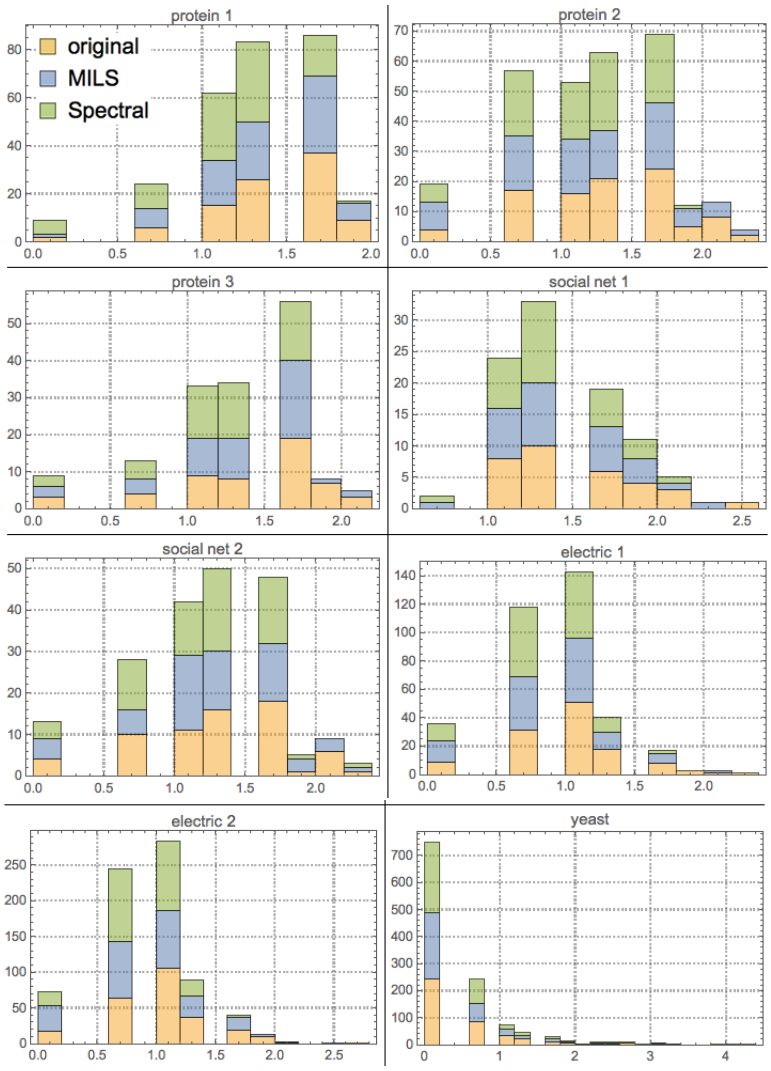

Figure 6: Stacked histograms showing the preservation of degree centrality after application of MILS versus spectral sparsification across different families of networks: bars with height closest to the original graph signify better preservation. Blue bars (MILS) approximate yellow (original) bars compared with spectral sparsification. MILS slightly outperforms spectral sparsification in this test but never did worse.

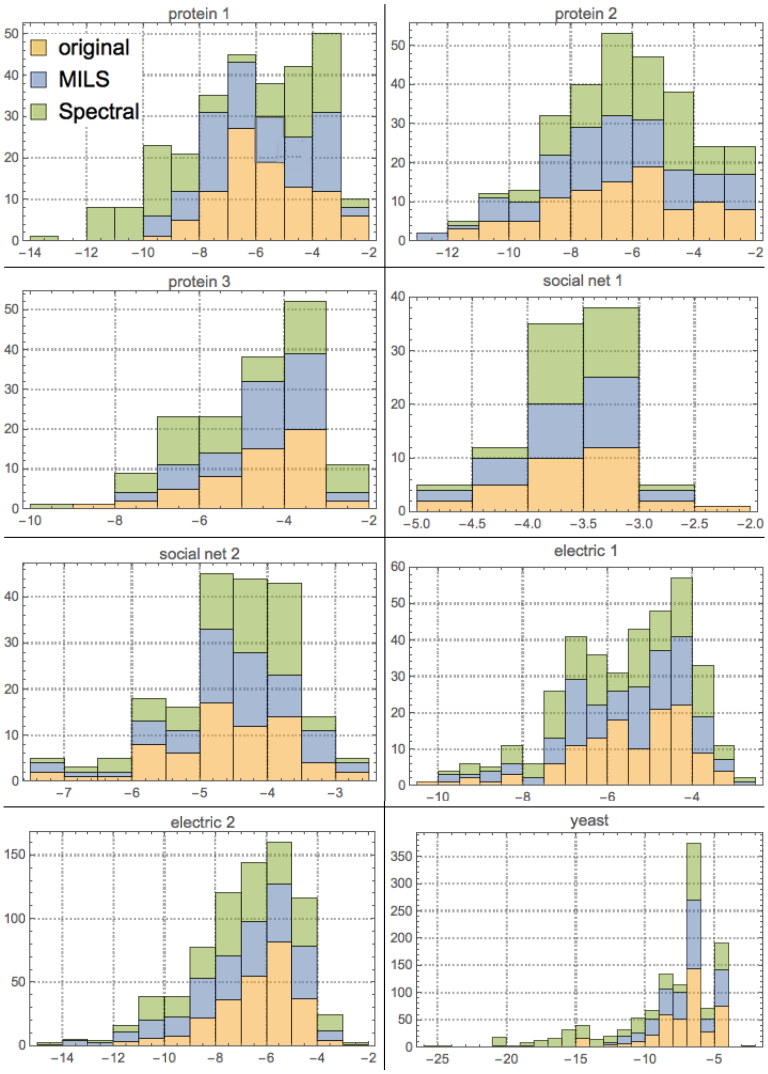

Figure 7: Stacked histograms showing eigenvector centrality preservation of MILS versus spectral sparsification across the different families of networks: bars with height closest to the graph's original bar signify better edge betweenness distribution preservation. Blue bars (MILS) approximate yellow (original) bars better than spectral sparsification both in distribution shape and individual bar height. On average MILS preserved the eigenvector centrality distribution of these representative networks 1.5 times better.