IndeCut evaluates performance of network motif discovery algorithms

Mitra Ansariola [1,2], Molly Megraw [1,2,3,*] and David Koslicki [1,4,*]

December 2, 2017

1 Center for Genome Research and Biocomputing, **2** Department of Botany and Plant Pathology, **3** Department of Computer Science, **4** Department of Mathematics, Oregon State University, Corvallis, OR 97331.

*To whom correspondence should be addressed. Email: megrawm@science.oregonstate.edu. Please address correspondence regarding sampling algorithms and their assessment in this work to Molly Megraw. Email: david.koslicki@math.oregonstate.edu : Please address correspondence regarding mathematics contained in this work to David Koslicki.

Method S1: Mathematical Details

In this section, we give the mathematical details necessary to support the claim that the cut norm and maximum entry matrix can be used to test for non-uniformity of a bipartite graph sampling algorithm. We begin by recalling the results of [1] that we use and then derive bounds necessary to estimate the cut norm.

Results from Barvinok [1]

Let $\Sigma(R,C)$ be the set of all binary matrices with row-sums $R=(r_1,\ldots,r_m)\in\mathbb{N}^m$ and column-sums $C=(c_1,\ldots,c_n)\in\mathbb{N}^n$. Let $\mathcal{P}(R,C)$ be the polytope of matrices with entries bounded between 0 and 1 and with row and column sums R and C respectively. Throughout, we only consider R and C such that for every choice of $1\leq i\leq m$ and $1\leq j\leq n$, there exist at least two matrices $L,M\in\Sigma(R,C)$ such that $L_{i,j}=0$ and $M_{i,j}=1$. This condition requires the space $\Sigma(R,C)$ to be reasonably large (i.e. the polytope $\mathcal{P}(R,C)$ is non-empty).

We now recount pertinent theorems from [1]. The first gives an estimate of the number of bipartite graphs with degree sequences R and C: $|\Sigma(R,C)|$.

Theorem 1 ([1, Theorem 1])**.** *Let*

$$F(\mathbf{x},\mathbf{y})=\left(\prod_{i=1}^{m}x_i^{-r_i}\right)\left(\prod_{j=1}^{n}y_j^{-c_j}\right)\left(\prod_{i,j}(1+x_iy_j)\right)$$

for $\mathbf{x}=(x_1,\ldots,x_m)$ *and* $\mathbf{y}=(y_1,\ldots,y_n)$*. Let*

$$\alpha(R,C)=\underset{\mathbf{x},\mathbf{y}>0}{\textit{minimum}}\quad F(\mathbf{x},\mathbf{y}).$$

Then

$$\alpha(R,C)\geq|\Sigma(R,C)|\geq\frac{(mn)!}{(mn)^{mn}}\left(\prod_{i=1}^{m}\frac{(n-r_i)^{n-r_i}}{(n-r_i)!}\right)\left(\prod_{j=1}^{n}\frac{c_j^{c_j}}{c_j!}\right)\alpha(R,C).$$

Taking the logarithm of $F(\mathbf{x},\mathbf{y})$ gives a convex function on $\mathbb{R}^{m\times n}$, so $\alpha(R,C)$ may be efficiently computed. This allows us to define the *maximum entropy matrix*:

Definition 1 ([1, Lemma 2])**.** *Let* $\mathbf{x}^*$ *and* $\mathbf{y}^*$ *be the vectors that obtain optimality in the definition of* $\alpha(R,C)$*. Define* $Z\in\mathbb{R}^{m\times n}$ *as*

$$Z_{i,j}=\frac{\mathbf{x}_i^*\mathbf{y}_j^*}{1+\mathbf{x}_i^*\mathbf{y}_j^*}. \tag{1}$$

The cut norm is needed to state the next theorem.

Definition 2. *Let* $A\in\mathbb{R}^{m\times n}$ *and let*

$$||A||_{\mathcal{C}}=\underset{\substack{I\subseteq\{1,\ldots,n\}\\J\subseteq\{1,\ldots,m\}}}{\textit{maximize}}\quad\left|\sum_{i\in I,j\in J}A_{i,j}\right|. \tag{Cut Norm}$$

Let $S\subset\{(i,j):i=1,\ldots,m,\ j=1,\ldots,n\}$ be a set of indicies. For a $m\times n$ matrix A, let

$$\sigma_S(A)=\sum_{(i,j)\in S}A_{ij}.$$

Note that $||A||_{\mathcal{C}}=\max_S|\sigma_S(A)|$.

We can now state the main result that serves as the justification of IndeCut. Recall our assumption that $|\Sigma(R,C)|\geq 2$ and so $\mathcal{P}(R,C)$ is non-empty.

Theorem 2 ([1, Theorem 3]). *Fix numbers $\kappa > 0$ and $0 < \delta < 1$ then there exists a number $q = q(\kappa, \delta)$ such that the following holds. Let R and C be such that $n \geq m > q$ and let $Z \in \mathcal{P}(R, C)$ be the maximum entry matrix. Let $S \subset \{(i,j) : i = 1, \ldots, m,\ j = 1, \ldots, n\}$ be such that $\sigma_S(Z) \geq \delta mn$ and let $\epsilon = \delta \frac{\ln n}{\sqrt{M}}$. If $\epsilon \leq 1$, then*

$$\mathbb{P}\left\{D \in \Sigma(R, C) :\ (1-\epsilon)\sigma_S(Z) \leq \sigma_S(D) \leq (1+\epsilon)\sigma_S(Z)\right\} \geq 1 - 2n^{-\kappa n}.$$

This theorem states that a uniformly sampled binary matrix is close to the maximum entry matrix in terms of the cut norm.

We now re-state this result in terms of the cut norm.

Theorem 3. *Let $Z \in \mathcal{P}(R, C)$ be the maximum entry matrix. Let $(A_i)_{i=1}^N$ be a sequence of independent and uniformly distributed random variables on $\Sigma(R, C)$ and let $A_{(N)} = \frac{1}{N}\sum_{i=1}^N A_i$. Let $S \subset \{(i,j) : i = 1, \ldots, m,\ j = 1, \ldots, n\}$ be such that $|\sigma_S(Z - A_{(N)})| = ||Z - A_{(N)}||_{\mathcal{C}}$. Let $0 < \delta < 1$ be such that for this S, $\sigma_S(Z) \geq \delta mn$ and let $\epsilon = \delta \frac{\ln n}{\sqrt{M}}$. Fix $\kappa > 0$, then there exists a number $q = q(\kappa, \delta)$ such that if R and C are such that $n \geq m > q$, the following holds: If $\epsilon \leq 1$, then*

$$\mathbb{P}\left(\frac{\left|\left|\frac{1}{N}\sum_{i=1}^N A_i - Z\right|\right|_{\mathcal{C}}}{||Z||_{\mathcal{C}}} \leq \epsilon\right) \geq 1 - 2n^{-\kappa n}. \tag{2}$$

Proof. Assuming that

$$(1-\epsilon)\sigma_S(Z) \leq \sigma_S\left(A_{(N)}\right) \leq (1+\epsilon)\sigma_S(Z),$$

since $||Z||_{\mathcal{C}} = \max_{S'} |\sigma_{S'}(Z)|$, this implies that

$$(1-\epsilon)||Z||_{\mathcal{C}} \leq \sigma_S\left(A_{(N)}\right) \leq (1+\epsilon)||Z||_{\mathcal{C}}. \tag{3}$$

By hypothesis, $|\sigma_S(Z - A_{(N)})| = ||Z - A_{(N)}||_{\mathcal{C}}$ and so along with linearity of $\sigma_S(\cdot)$, equation (3) implies that

$$||A_{(N)} - Z||_{\mathcal{C}} \leq \epsilon||Z||_{\mathcal{C}}.$$

Monotonicity of probability and the conclusion of Theorem 2 then imply that

$$\mathbb{P}\left(\frac{\left|\left|\frac{1}{N}\sum_{i=1}^N A_i - Z\right|\right|_{\mathcal{C}}}{||Z||_{\mathcal{C}}} \leq \epsilon\right) \geq 1 - 2n^{-\kappa n}.$$

□

Given appropriate R, C, κ, δ, and ϵ, the contrapositive of this result implies that if $\frac{\left|\left|A_{(N)} - Z\right|\right|_{\mathcal{C}}}{||Z||_{\mathcal{C}}}$ is large, then there is an exponentially small chance that the sequence of random variables is independent and uniformly distributed. This is the justification to use the quantity

$$\frac{\left|\left|A_{(N)} - Z\right|\right|_{\mathcal{C}}}{||Z||_{\mathcal{C}}}$$

as a measure of non-uniformity/independence and forms the mathematical justification of IndeCut. We turn now to looking at how to calculate this quantity in practice.

Computing norms

The cut norm $||\cdot||_{\mathcal{C}}$ is difficult to compute (in fact, it is MAX SNP hard [2]) for general matrices, but we will be able to relate it to another norm ($||\cdot||_{\infty \mapsto 1}$) that can be approximated with a semidefinite relaxation. We then round the solution of the semidefinite relaxation to get an estimate of $||\cdot||_{\infty \mapsto 1}$ and hence of $||\cdot||_{\mathcal{C}}$. We begin with definitions of the norms of interest.

Definition 3. *Let $A \in \mathbb{R}^{m\times n}$. Define the following norms by*

$$||A||_{\infty\mapsto 1} = \underset{\substack{x_i\in\{-1,+1\}\\ y_j\in\{-1,+1\}}}{\textit{maximize}} \quad \sum_{i,j} A_{i,j}x_iy_j \qquad (\infty \mapsto 1 \text{ Norm})$$

We denote the semidefinite relaxation of $||A||_{\infty\mapsto 1}$ by $||A||_{\text{SDR}}$:

$$||A||_{\text{SDR}} = \underset{||u_i||_2=||v_j||_2=1}{\text{maximize}} \quad \sum_{i,j} A_{i,j}(u_i\cdot v_j) \qquad (\text{SDR Norm})$$

Note that $||A||_{\text{SDR}}$ can be converted to the following optimization problem:

$$\begin{aligned} ||A||_{\text{SDR}} = \frac{1}{2}\underset{X}{\text{maximize}} \quad & \text{tr}(CX) \\ \text{subject to} \quad & \text{tr}(F_kX) = a_k,\ k = 1,\dots,m+n \\ & X \succeq 0, \end{aligned} \tag{4}$$

for

$$C = \begin{bmatrix} 0 & A \\ A & 0 \end{bmatrix},\ F_k = \begin{cases} 1 & \text{if } i=j=k \\ 0 & \text{o.w.} \end{cases},\ \text{and } a_k = 1 \text{ for } k = 1,\dots,m+n.$$

This form allows us to use popular computational packages to compute $||\cdot||_{\text{SDR}}$. We utilize the computational package CSDP version 6.1.0 [3].

It turns out that for the matrices of interest, the norms $||\cdot||_C$ and $||\cdot||_{\infty\mapsto 1}$ are equal up to a factor of 4. Indeed, note the maximum entropy matrix Z defined in equation (1) and $A_{(N)}$ defined in the previous section both have row/column sums equal to R and C: $A_{(N)}, Z \in \Sigma(R,C)$. Hence the matrix $Z - A_{(N)}$ has zero row and column sum. This allows us to obtain the well-known [2, 4] relationship between the norms $||\cdot||_{\infty\mapsto 1}$ and $||\cdot||_{\mathcal{C}}$.

Proposition 4. *If the matrix A has zero row and column sums (i.e. $\sum_i A_{i,j} = \sum_j A_{i,j} = 0$), then $||A||_{\infty\mapsto 1} = 4||A||_{\mathcal{C}}$.*

Proof. For $I \subseteq \{1,\dots,n\}$ and $J \subseteq \{1,\dots,m\}$ the sets achieving the maximum in the definition of $||A||_{\mathcal{C}}$, define $x_i = 1$ for $i \in I$, $x_i = -1$ for $i \notin I$ and $y_j = 1$ for $j \in J$, $y_j = -1$ for $j \notin J$. Then

$$\begin{aligned} ||A||_{\mathcal{C}} &= \sum_{i,j} A_{i,j}\frac{1+x_i}{2}\frac{1+y_j}{2} \\ &= \frac{1}{4}\left(\sum_{i,j}A_{i,j} + \sum_{i,j}A_{i,j}x_i + \sum_{i,j}A_{i,j}y_j + \sum_{i,j}A_{i,j}x_iy_j\right) \\ &= \frac{1}{4}\sum_{i,j}A_{i,j}x_iy_j \\ &= \frac{1}{4}||A||_{\infty\mapsto 1}. \end{aligned}$$

□

Cut norm estimates

In [2, Section 5.1], an algorithm was presented that computes bounds on $||\cdot||_{\infty\mapsto 1}$. We use a slight modification of this algorithm that gives tighter bounds in practice as follows:

Given a matrix A, let $u_i, v_j \in \mathbb{R}^{m+n}$, for $i = 1,\dots,m, j = 1,\dots,n$ be the optimal vectors obtained from the computation of $||A||_{\text{SDR}}$. Let $g_i \sim \mathcal{N}(0,1)$, $i = 1,\dots,m+n$, be independent standard normal random variables and let $G = (g_1,\dots,g_{m+n})$. Let $x_i = \text{sign}(u_i\cdot G)$ and $y_j = \text{sign}(v_j\cdot G)$.

Now $\sum_{i,j} A_{i,j}x_iy_j \leq ||A||_{\infty\mapsto 1}$ since $||A||_{\infty\mapsto 1}$ is the maximum value. However, there is a positive probability that $\sum_{i,j} A_{i,j}x_iy_j = ||A||_{\infty\mapsto 1}$. To observe this fact, let $x_i^*, y_j^* \in \{-1, +1\}$ be such that $||A||_{\infty\mapsto 1} = \sum_{i,j} A_{i,j}x_i^*y_j^*$. We can find at least one vector G^* such that $x_i^* = \text{sign}(u_i \cdot G^*)$ and $y_j^* = \text{sign}(v_j \cdot G^*)$ since this reduces to solving a solvable system of linear inequalities due to the u_i, v_j being obtained from eigenvectors of the spectral factorization of X in the optimization procedure (4). Given such a G^*, note that for any $a \in \mathbb{R}$, $a > 0$, $x_i^* = \text{sign}(u_i \cdot aG^*)$ and $y_j^* = \text{sign}(v_j \cdot aG^*)$. Hence, with probability at least 2^{-m-n}, a randomly chosen G will result in obtaining the optimal x_i^* and y_j^*. We do not attempt to make a more nuanced estimation of this probability since only bounds are necessary for our purposes.

Repeating the above rounding procedure a number of times and taking the maximum result, we obtain Algorithm 1 which we use to compute the bounds on the cut norm of a matrix A. In practice, we take the number of iterates of Algorithm 1 to be $1,000$. Denote the output of this algorithm with $||A||^{\text{est}}_{\infty\mapsto 1}$. As a result, $||A||^{\text{est}}_{\infty\mapsto 1} \leq ||A||_{\infty\mapsto 1} \leq ||A||_{\text{SDR}}$, so combining these with proposition 4, we have the following estimation of the cut norm:

$$\frac{1}{4}||A||^{\text{est}}_{\infty\mapsto 1} \leq ||A||_{\mathcal{C}} \leq \frac{1}{4}||A||_{\text{SDR}}. \tag{5}$$

We apply this estimation to obtain bounds on the quantity of interest:

$$\frac{\left|\left|A_{(N)} - Z\right|\right|_{\mathcal{C}}}{4||Z||_{\mathcal{C}}}. \tag{6}$$

We can compare motif finding algorithms in the following fashion: Let $(A_i)_{i=1}^N$ and $(B_i)_{i=1}^N$ be N random binary matrices generated by two algorithms $\mathcal{A}$ and $\mathcal{B}$. If the upper bound for one algorithm (say, $\mathcal{A}$) falls below the lower bound of the other algorithm (say, $\mathcal{B}$), then we can be sure that the cut norm quantity of interest for $\mathcal{A}$ is smaller than for $\mathcal{B}$. As a consequence of the previous section, this implies that we can be more confident that $\mathcal{B}$ samples the space in a less uniform and independent fashion. If the bounds do not overlap, then no conclusion can be made since we cannot guarantee that one cut norm is larger than the other. More rigorously, let $A_{(N)} = \frac{1}{N}\sum_{i=1}^N A_i$ and $B_{(N)} = \frac{1}{N}\sum_{i=1}^N B_i$. If for sufficiently large N, we have that

$$||A_N - Z||_{\text{SDR}} < ||B_{(N)} - Z||^{\text{est}}_{\infty\mapsto 1}$$

then equation 5 implies that

$$\frac{\left|\left|A_{(N)} - Z\right|\right|_{\mathcal{C}}}{4||Z||_{\mathcal{C}}} < \frac{\left|\left|B_{(N)} - Z\right|\right|_{\mathcal{C}}}{4||Z||_{\mathcal{C}}}.$$

As a consequence of Theorem 3 and the discussion that followed, we can be confident that $\mathcal{B}$ samples the space $\Sigma(R, C)$ in a less uniform and independent fashion than $\mathcal{A}$. The symmetric case of overlapping bounds $||B_N - Z||_{\text{SDR}} < ||A_{(N)} - Z||^{\text{est}}_{\infty\mapsto 1}$ would imply the reverse conclusion being made about $\mathcal{A}$ and $\mathcal{B}$. If, however, the bounds overlap:

$$\left(\frac{||Z - A_{(N)}||^{\text{est}}_{\infty\mapsto 1}}{4||Z||_{\mathcal{C}}}, \frac{||Z - A_{(N)}||_{\text{SDR}}}{4||Z||_{\mathcal{C}}}\right) \cap \left(\frac{||Z - B_{(N)}||^{\text{est}}_{\infty\mapsto 1}}{4||Z||_{\mathcal{C}}}, \frac{||Z - B_{(N)}||_{\text{SDR}}}{4||Z||_{\mathcal{C}}}\right) \neq \emptyset,$$

then no conclusion can be drawn as no information is provided about the relative sizes of the cut norms. Hence, we use the quantity:

$$\mathsf{IndeCut}(Z, \mathcal{A}, N) = \left(\frac{||Z - A_{(N)}||^{\text{est}}_{\infty\mapsto 1}}{4||Z||_{\mathcal{C}}}, \frac{||Z - A_{(N)}||_{\text{SDR}}}{4||Z||_{\mathcal{C}}}\right)$$

to compare uniformity/independence of motif finding algorithms.

Algorithm 1 : Cut norm lower bound

Input:

$A \in \mathbb{R}^{m \times n}$

$c \in \mathbb{N}$

$u_i, v_j \in \mathbb{R}^{m+n}$ (from computation of $||A||_{\text{SDR}}$)

Initialization:

$its = 0$

$bound = 0$

Iterations:

while $its < c$ **do**

$G = (g_1, \ldots, g_{m+n})$ (random variates of $\mathcal{N}(0, 1)$)

for $i = 1, \ldots, m$ **do**

$x_i = \text{sign}(u_i \cdot G)$

end for

for $j = 1, \ldots, n$ **do**

$y_j = \text{sign}(y_j \cdot G)$

end for

$temp = \sum_{i,j} A_{i,j} x_i, y_j$

if $temp > bound$ **then**

$bound = temp$

end if

$its = its + 1$

end while

Output:

$||A||^{\text{est}}_{\infty \mapsto 1} = bound$ (Lower bound)

Method S2: Description of examined network motif discovery algorithms

In order to compare the performance of existing network motif discovery algorithms using IndeCut, four different network motif finding algorithms were selected: FANMOD (Fast Network Motif Detection) [5], DIA-MCIS (Diaconis Monte Carlo Importance Sampling) [6], WaRSwap (Weighted and Reverse Swap sampling) [7], and CoMoFinder [8].

FANMOD is a well-known implementation of the edge switching randomization algorithm. The edge-switching method randomly chooses two directed edges (x, y), (u, v) from input graph G and switches their endpoints only if G doesn't already contain either of these new edges (x, v), (u, y). It repeats this procedure for defined number of attempts and reports a random graph G'. An implementation of FANMOD was downloaded from [5] and we added a print statement in the source code "main.cpp" which prints the edges of the randomized graph produced by the method named "randomized_graph" so we can read them as input for IndeCut.

CoMoFinder implements a restricted version of the edge-switching method to detect only K-node motifs containing all node types such as TF, miRNA, and Gene, on given TF-miRNA-Gene regulatory networks. It breaks down the original network into seven different layers (miRNA $\rightarrow$ TF, TF $\rightarrow$ gene, miRNA TF, TF TF, TF $\rightarrow$ miRNA, TF $\rightarrow$ TF, TF $\rightarrow$ gene). Within each layer it chooses two edges (x, y), (u, v) and switches the endpoints if two conditions satisfied: 1) Neither of the new edge-pairs (x, v), (u, y) exist in the input graph G, and 2) An edge-switch between (x, y), (u, v) is allowed to happen only once, as revisiting a previously performed switch is not allowed (i.e. switching back from a graph containing (x, v) and (u, y) to a graph containing (x, y) and (u, v) is not allowed). CoMoFinder repeats the above-described procedure until either it reaches a stage such that no edge-pair is available to switch, or it has completed a pre-defined maximum number of edge-switching attempts. The original CoMoFinder program [8] was downloaded and modified to print randomized graphs into files for our analysis.

DIA-MCIS is an efficient implementation of an importance sampling algorithm [9] to generate random graphs (self-loops included) from fixed in/out-degree sequences. DIA-MCIS converts an input graph G into a zero-one adjacency matrix M_{m*n} with m rows and n columns where m_{ij} is 1 if node j has a directed link to node i. It then sequentially fills the columns by a weighted-sampling scheme. It starts with first column which represents the first source node with out-degree of $\deg_0$, and assigns $\deg_0$ 1s randomly to m cells (each cell represents a target node). In this process, nodes with higher in-degrees have more chance of selection by source nodes with higher out-degrees. The algorithm updates the row/column sums as proceeds to the next column.

WaRSwap produces randomized background graphs from an input graph by breaking it into layers representing five possible interaction types: TF$\rightarrow$TF, TF$\rightarrow$miRNA, TF$\rightarrow$gene, miRNA$\rightarrow$TF, and miRNA$\rightarrow$gene. WaRSwap treats each layer as a bipartite graph G and operates as follows to generate a randomized graph G'. It first sorts the source nodes in descending order of out-degree, and for each source node S_i it computes the sampling weights for each target node T_j using a weighting formula [7]. The weighting formula corrects the tendency of source nodes with large out-degrees to target nodes with larger in-degrees. WaRSwap places an edge between S_i and T_j if possible, otherwise it enters a specific back-swapping procedure to identify a new target node. We downloaded a Java implementation of WaRSwap from `http://megraw.cgrb.oregonstate.edu/software/WaRSwapSoftwareApplication/` and R implementation from `http://megraw.cgrb.oregonstate.edu/software/WaRSwap`. The WaRSwapApp makes an automated selection of the WaRSwap weighting parameter for the user based on the in/out-degree sequences of the input graph. We modified the R implementation of WaRSwap to include this automated weighting parameter selection.

Method S3: Compute Relationship Between Number of Samples and Cut norm Estimates

Given the space of all sampled graphs produced by an algorithm $\{G_1, \ldots, G_n\}$, we generated m sets of samples $\{S_1, \ldots, S_m\}$ in which the set S_1 contained the first 100 sample graphs $\{G_1, \ldots, G_{100}\}$,

set S_2 contained all of the samples from S_1 plus the next 100 samples $\{G_{101}, \ldots, G_{200}\}$, and so on, until S_m contained all of the sample graphs $\{G_1, \ldots, G_n\}$. We then used IndeCut to compute cut norm estimates for each set of subsamples S_i, in order to identify an approximate sample size at which the cut norm estimate for S_i became very close to the cut norm estimate for the entire sample space $S_m = \{G_1, \ldots, G_n\}$. Fig. S13 shows a visualization of the relationship between the number of samples and the cut norm estimates for a large biological network (TF $\rightarrow$ Gene network extracted from the Human regulatory network).

In order to help user to estimate a sensible cutoff range for required number of samples for each algorithm and network we provide a visualization plugin to the IndeCut software package. This plugin creates plots that help the user to visualize the relationship between the number of samples and cutnorm estimates (see IndeCut's User Manual for details: `https://github.com/megrawlab/IndeCut/blob/master/README.md`) and allows the user to choose a number of samples corresponding to a point where the cut-norm is decreasing slowly enough for her/his application. For the programs and graphs considered in the manuscript, we have observed that 2500 samples would typically be a conservative estimate on an effective number of iterations. In general, an estimate of the number of samples required to achieve 'optimal' sampling performance varies with respect to network motif discovery programs and input graphs. When computing power is an issue and the user wishes to determine a 'minimum' number of iterations necessary for their network and method of interest, IndeCut's visualization plugin provides direct access to such plots for making this judgment call.

Method S4: Networks and graphs

Two sets of graphs were created or selected for this study: 1) Manually constructed "toy" bipartite graphs with sizes ranging from tens of nodes to hundreds of nodes, representing different graph structures, including "even" or "near-even" graphs, "uneven" graphs, and "hybrid" combinations of in/out-degrees, and 2) Real biological networks.

Real networks - Two biological networks were obtained from literature and public databases. An *Ecoli* network representing a medium-size yeast transcriptional network was downloaded from [10]. This network contains two types of nodes: transcription factor (TF), and gene. Two layers of interactions (TFgene, TFTF) were extracted into separate bipartite graphs for application of IndeCut. A *Human* regulatory network was downloaded from `http://encodenets.gersteinlab.org/`, representing a network with thousands of nodes and edges. This network is used as a case study in the publication of CoMoFinder [8]. This network contains three types of nodes: TF, miRNA, and protein-coding gene. This network comprises five interaction layers: TFTF, TFmiRNA, miRNATF, TFgene, and miRNA-gene. Each of these layers forms a separate input bipartite graph for IndeCut.

Method S5: Description of edge switch graphs

We detail here how the edge switch graphs (ESG's) were created. Given in and out-degrees R and C, we generate all possible bipartite graphs $\{G_1, \ldots, G_N\}$ with in/out-degrees R and C. The edge switch graph G_{ESG} is an undirected graph with vertex set $V = \{G_1, \ldots, G_N\}$ and edge set E defined as follows: for $G_i, G_j \in V$, the undirected edge (G_i, G_j) is an element of E if and only if the graph G_j can be obtained as a result of performing one edge switch on G_i. In more detail, this means that the graphs G_i and G_j have the same vertex set, and identical edge sets, except for one pair of edges (x, y) and (u, v) present in the edge set of G_i but absent in the edge set of G_j, and one pair of edges (x, v) and (u, y) present in the edge set of G_j but absent in the edge set of G_i.

A graph clustering algorithm known as modularity clustering [11] was then applied to the edge switch graph G_{ESG} to identify clusters that maximize the number of within-cluster edges while minimizing the number of between-cluster edges. Let L be the number of clusters found.

Given a graph sampling algorithm $\mathcal{A}$, the output of $\mathcal{A}$ can be viewed as sampling vertices of the ESG. Define a count vector $count^{\mathcal{A}} \in \mathbb{N}^L$ as a vector indexed by the clusters found above, with $count_i^{\mathcal{A}}$ being equal to the number of times the algorithm $\mathcal{A}$ returned a graph found in cluster i.

A "cluster-time" graph is then created with vertices corresponding to the clusters found above, and edges between two pairs of vertices/clusters if there exists edges in G_{ESG} connecting vertices belonging to these two clusters respectively. The size of the vertex i corresponds to the entry of the count vector $count_i^{\mathcal{A}}$. The entropy of the vector $count^{\mathcal{A}}$ is also calculated to quantify how equally (or unequally) the algorithm $\mathcal{A}$ samples graphs belonging to each cluster: $-\sum_{i=1}^{L} \frac{count_i^{\mathcal{A}}}{\sum_j count_j^{\mathcal{A}}} \log\left(\frac{count_i^{\mathcal{A}}}{\sum_j count_j^{\mathcal{A}}}\right)$. Larger entropy values indicate that the algorithm $\mathcal{A}$ samples each cluster more equally.

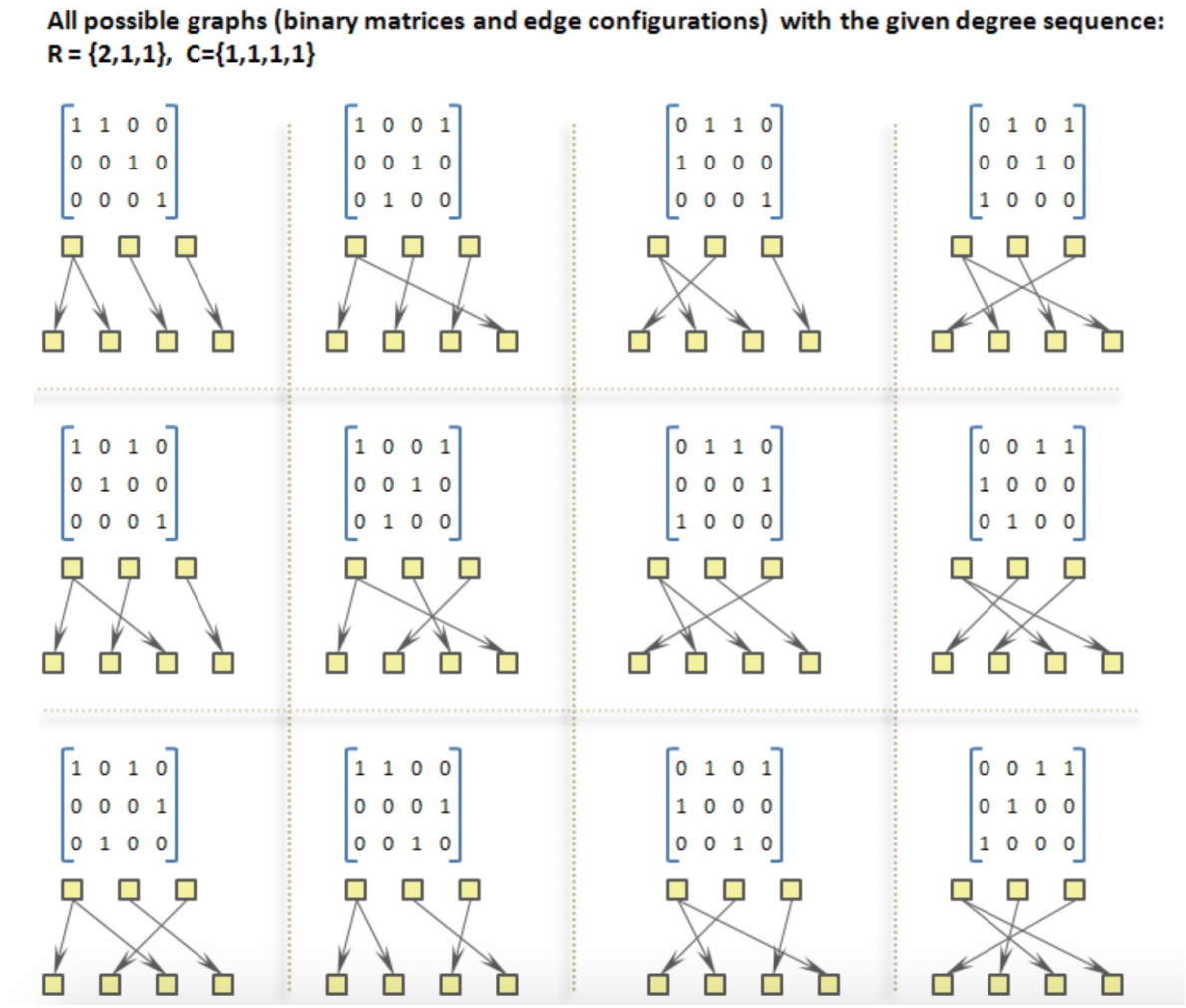

Figure S1: Sample space of an example degree sequence.The sample space of this degree sequence contains 12 different graphs.

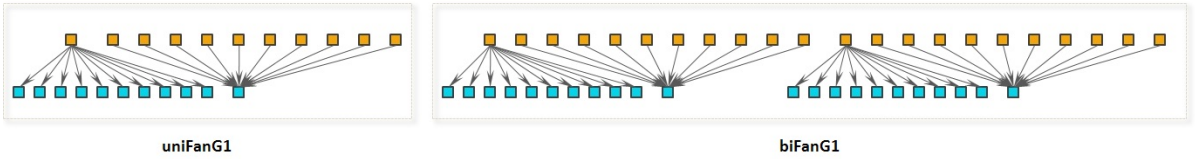

Figure S2: Constructing multiFan graphs starting from uniFanG1. A biFan graph is created by attaching two uniFanG1 graphs.

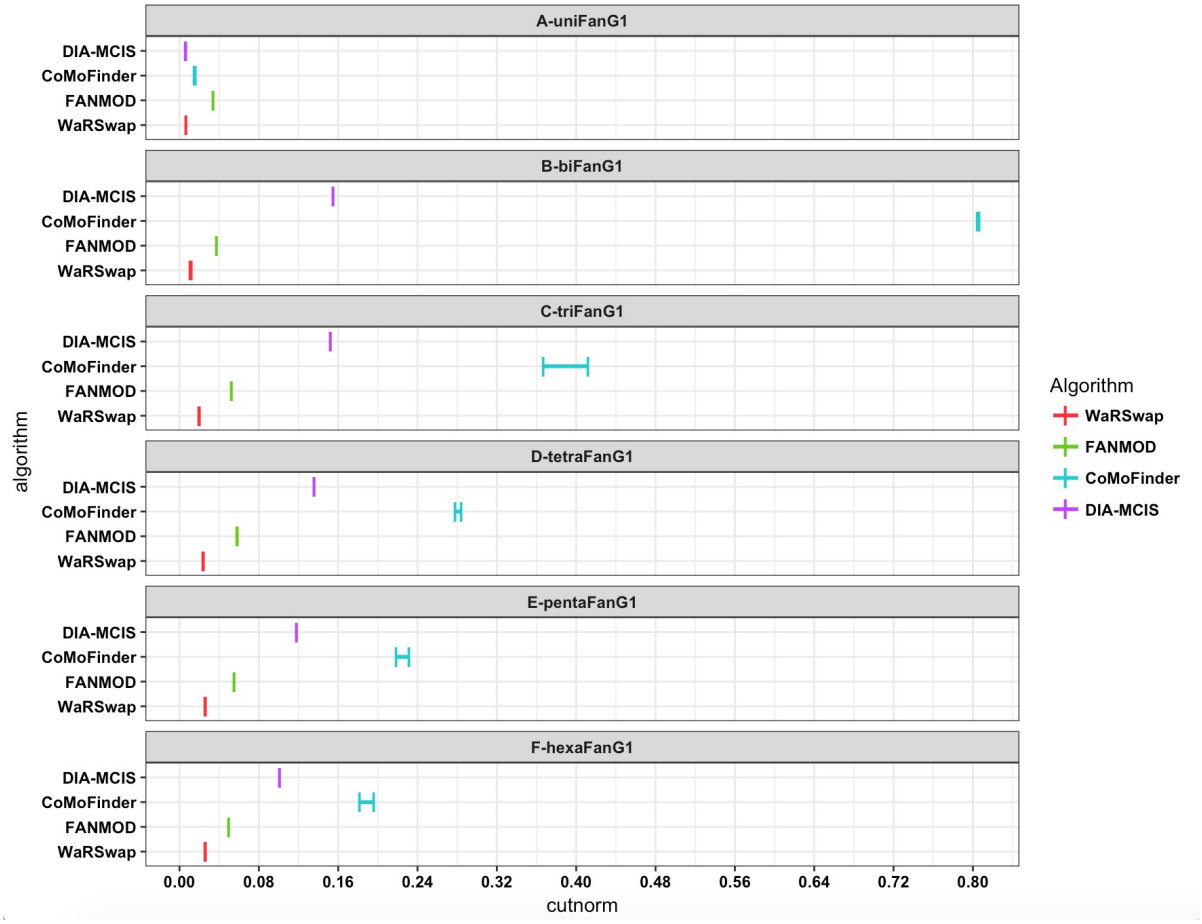

Figure S3: Graph sampling performance evaluation on small uneven graphs using IndeCut. This figure shows the cut norm estimates for all four examined algorithms: WaRSwap, CoMoFinder, DIA-MCIS, and FANMOD. For each graph and algorithm, 5000 graphs were generated. The cut norm estimates for each algorithm were computed using IndeCut. The vertical lines represent lower and upper bounds returned by the cut norm estimation with the true (NP-hard) value lying in this interval. A cut norm interval that is far from zero represents less uniform and independent sampling.

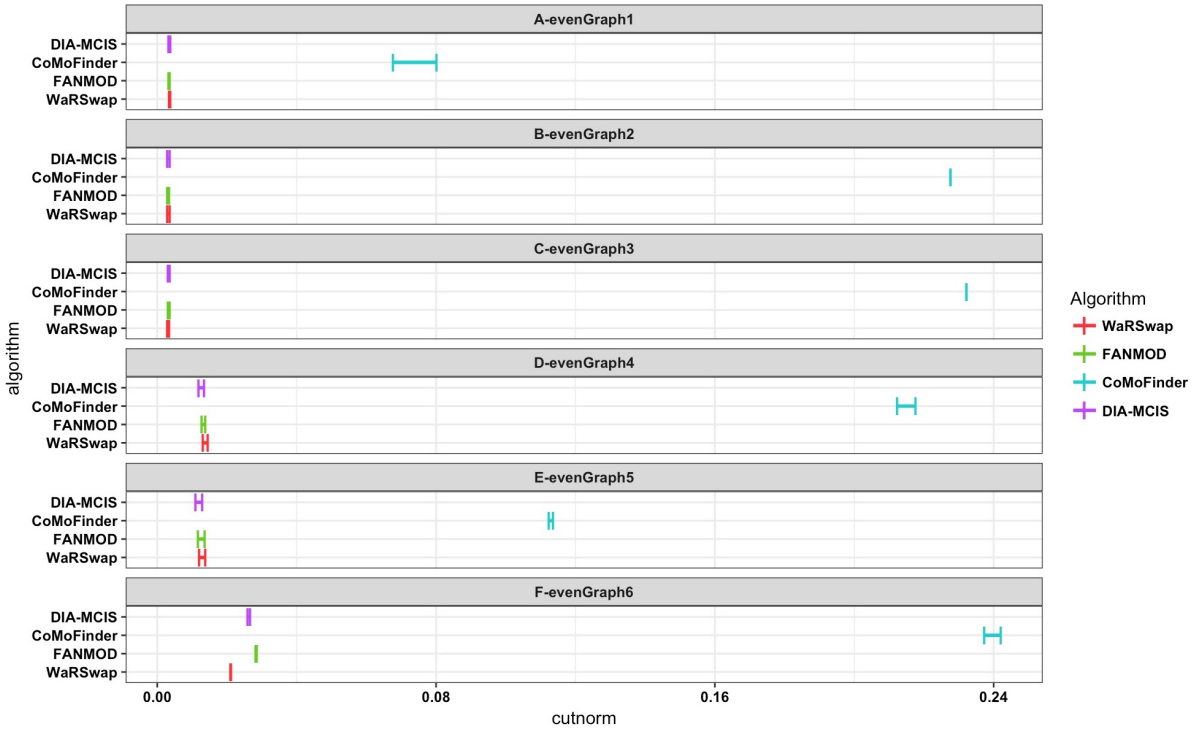

Figure S4: Graph sampling performance evaluation on small even graphs using IndeCut. This figure shows the cut norm estimates for all four examined algorithms: WaRSwap, CoMoFinder, DIA-MCIS, and FANMOD. For each graph and algorithm, 5000 graphs were generated. The cut norm estimates for each algorithm were computed using IndeCut. The vertical lines represent lower and upper bounds returned by the cut norm estimation with the true (NP-hard) value lying in this interval. A cut norm interval that is far from zero represents less uniform and independent sampling.

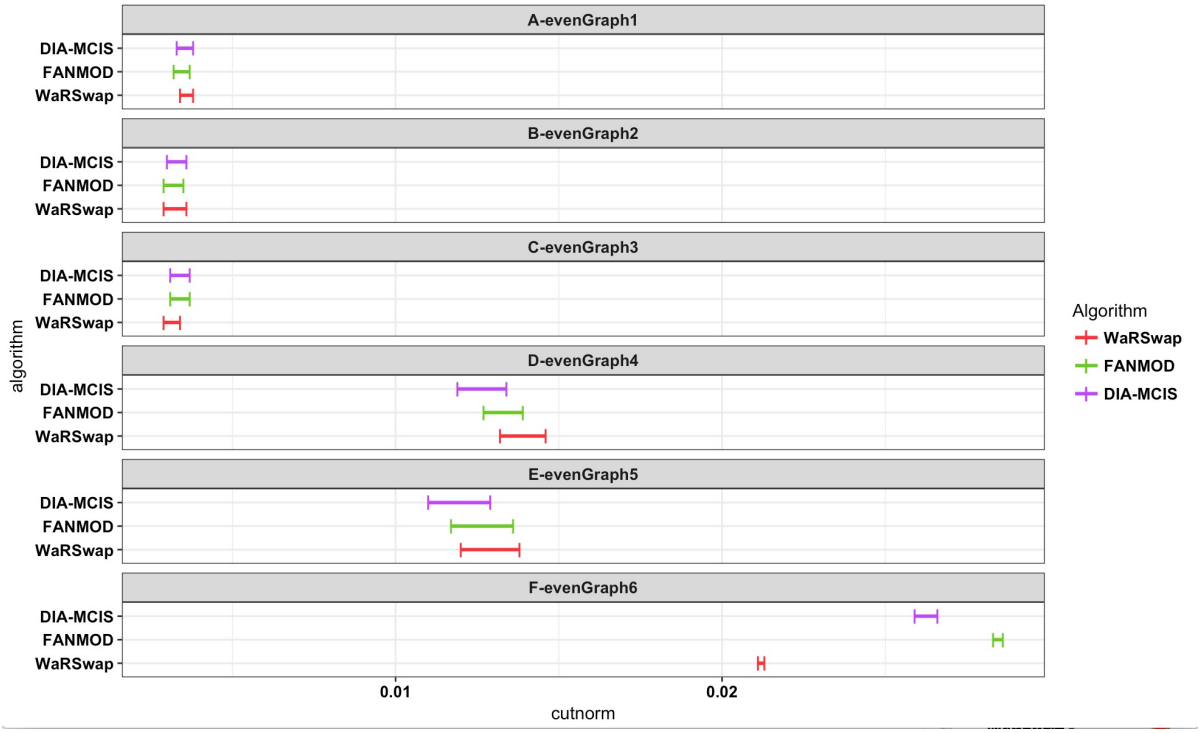

Figure S5: Zoomed view of uniform/independent graph sampling performance evaluation on small even graphs. For each small even graph and algorithm, 5000 graphs were generated. The cut norm estimates for each algorithm were computed using IndeCut. The vertical lines represent lower and upper bounds returned by the cut norm estimation with the true (NP-hard) value lying in this interval. A cut norm interval that is far from zero represents less uniform and independent sampling. The cut norm estimates for CoMoFinder were much larger than 0.06, therefore we removed CoMoFinder's results from this figure for ease of comparison (see Table S1 and Figure S4 for detailed results).

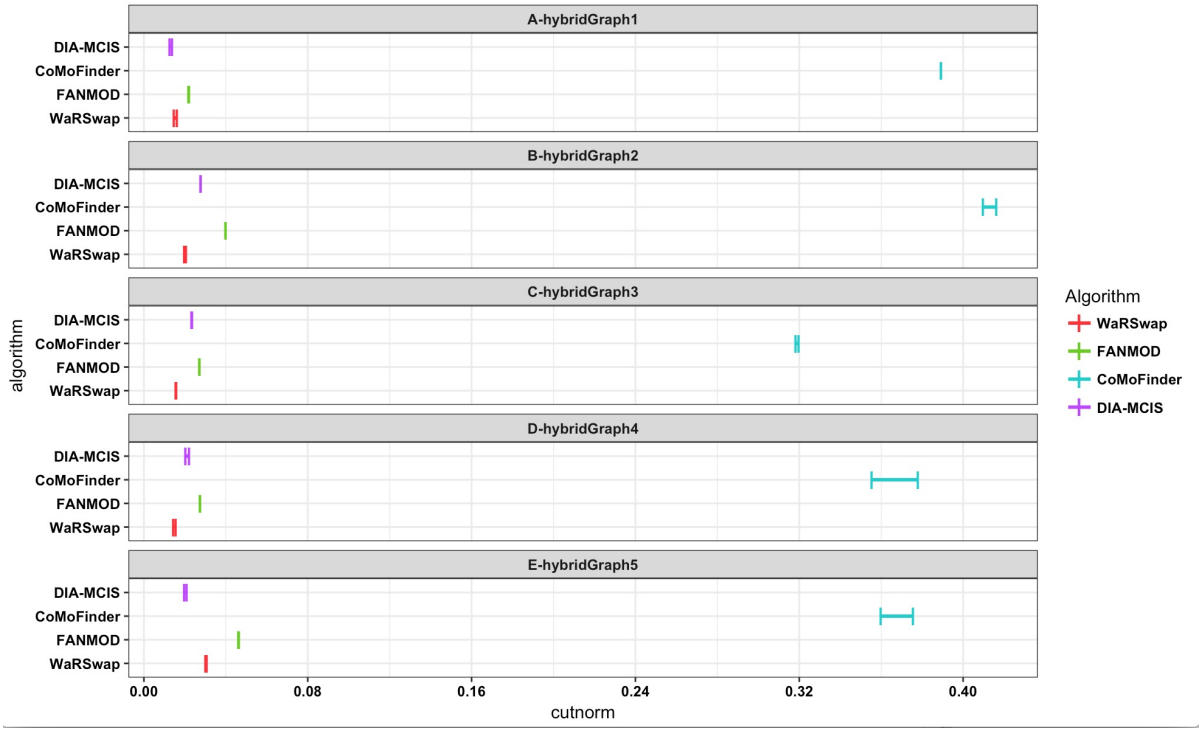

Figure S6: Graph sampling performance evaluation on small hybrid graphs using IndeCut. This figure shows the cut norm estimates for all four examined algorithms: WaRSwap, CoMoFinder, DIA-MCIS, and FANMOD. For each graph and algorithm, 5000 graphs were generated. The cut norm estimates for each algorithm were computed using IndeCut. The vertical lines represent lower and upper bounds returned by the cut norm estimation with the true (NP-hard) value lying in this interval. A cut norm interval that is far from zero represents less uniform and independent sampling.

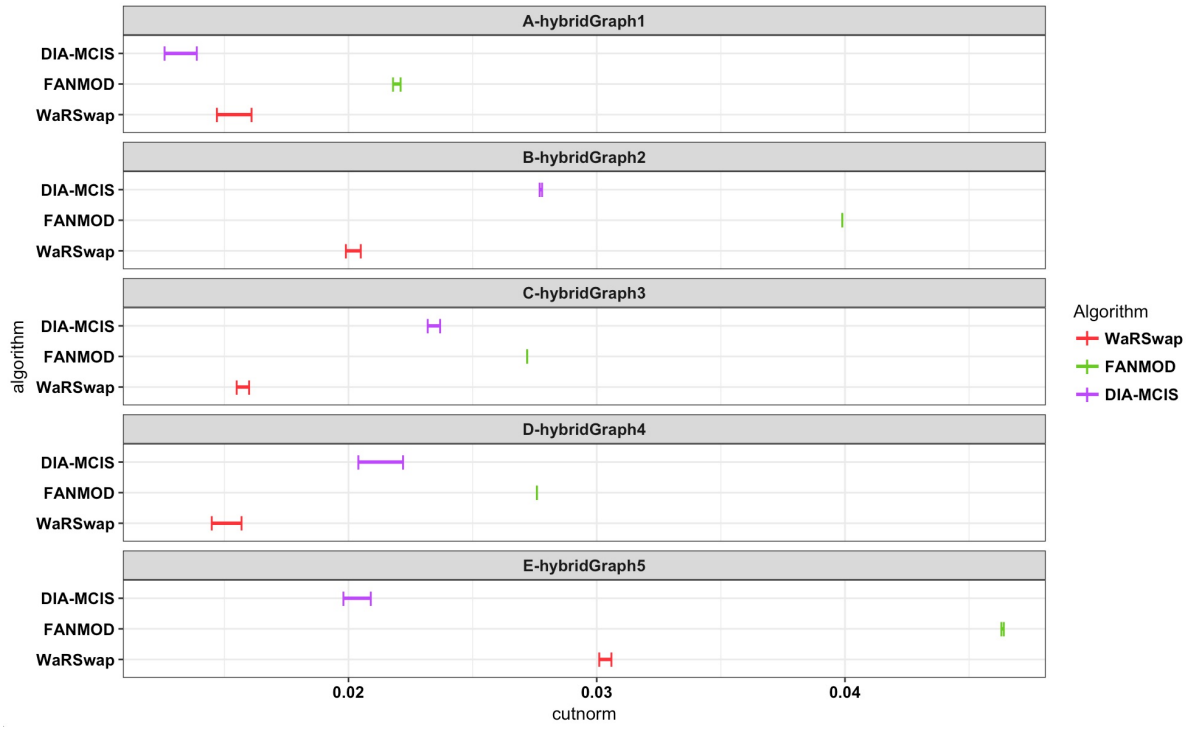

Figure S7: Zoomed view of uniform/independent graph sampling performance evaluation on small hybrid graphs. For each small even graph and algorithm, 5000 graphs were generated. The cut norm estimates for each algorithm were computed using IndeCut. The vertical lines represent lower and upper bounds returned by the cut norm estimation with the true (NP-hard) value lying in this interval. A cut norm interval that is far from zero represents less uniform and independent sampling. The cut norm estimates for CoMoFinder were much larger than 0.04, therefore we removed CoMoFinder's results from this figure for ease of comparison (see Table S1 and Figure S6 for detailed results).

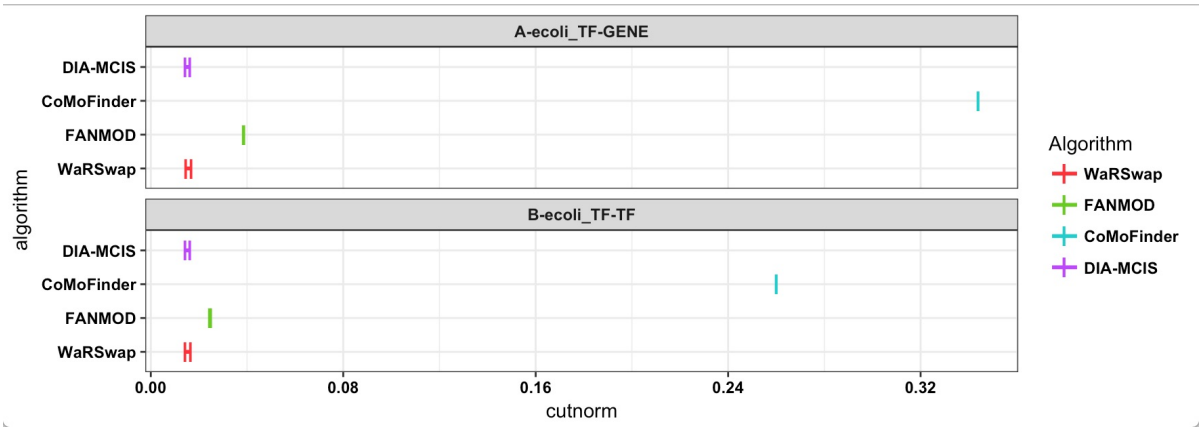

Figure S8: Graph sampling performance evaluation on Ecoli network using IndeCut. This figure shows the cut norm estimates for all four examined algorithms: WaRSwap, CoMoFinder, DIA-MCIS, and FANMOD. For each graph and algorithm, 5000 graphs were generated. The cut norm estimates for each algorithm were computed using IndeCut. The vertical lines represent lower and upper bounds returned by the cut norm estimation with the true (NP-hard) value lying in this interval. A cut norm interval that is far from zero represents less uniform and independent sampling.

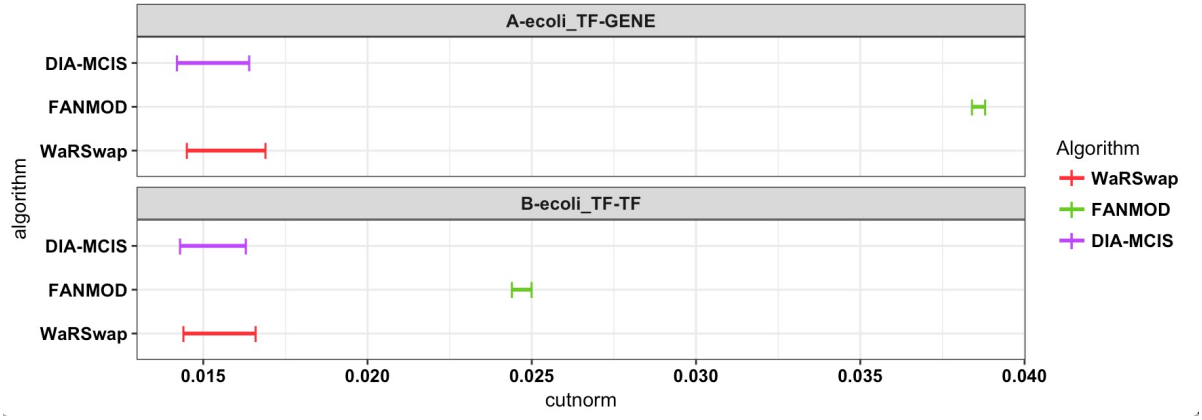

Figure S9: Zoomed view of uniform/independent graph sampling performance evaluation on the Ecoli regulatory network. For each small even graph and algorithm, 5000 graphs were generated. The cut norm estimates for each algorithm were computed using IndeCut. The vertical lines represent lower and upper bounds returned by the cut norm estimation with the true (NP-hard) value lying in this interval. A cut norm interval that is far from zero represents less uniform and independent sampling. (A) Cut norm bounds resulting from running IndeCut on the Ecoli TF→TF network. (B) Cut norm bounds resulting from running IndeCut on the Ecoli TF→Gene network. The cut norm estimates for CoMoFinder were much larger than 0.04, therefore we removed CoMoFinder's results from this figure for ease of comparison (see Table S1 and Figure S8 for detailed results).

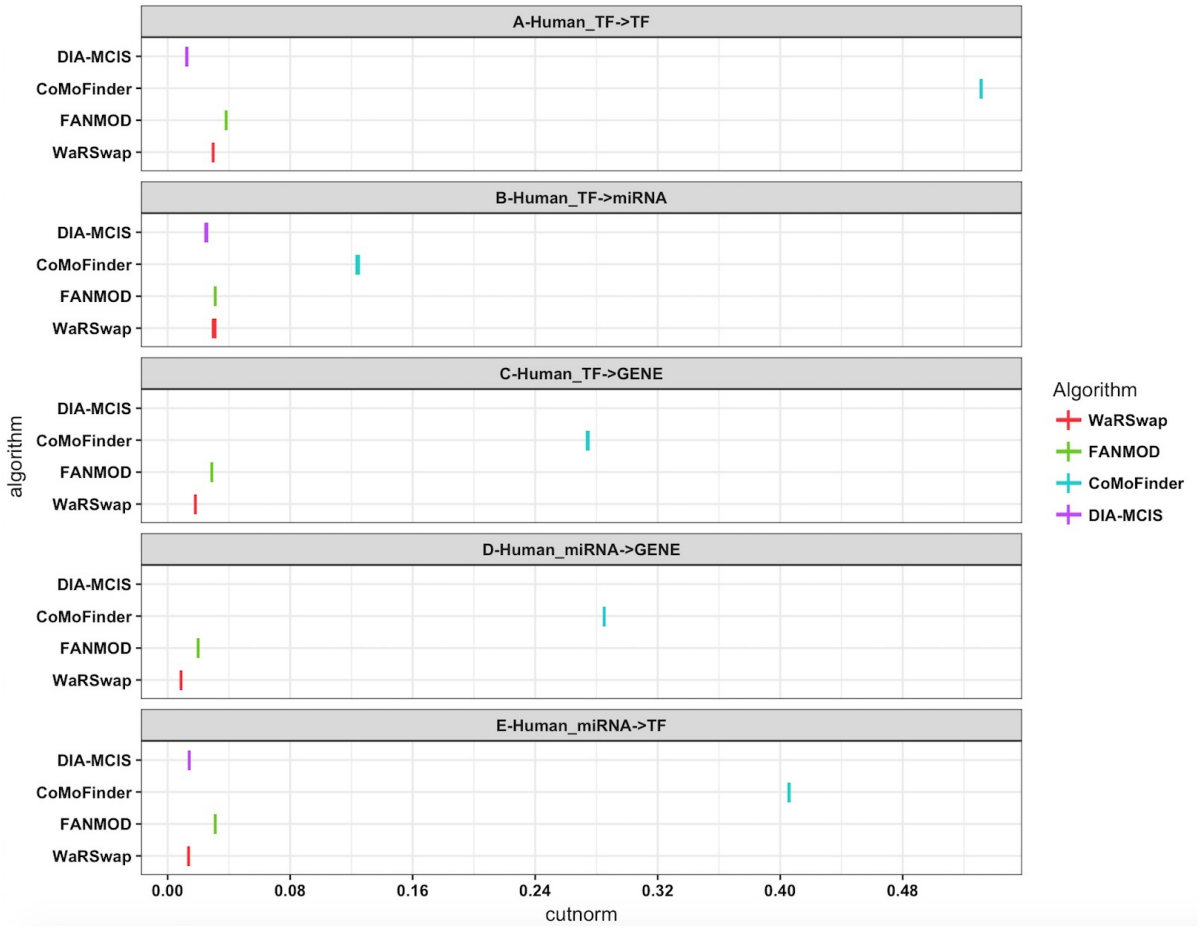

Figure S10: Graph sampling performance evaluation on Human regulatory network using IndeCut. The vertical lines represent lower and upper bounds returned by the cut norm estimation with the true (NP-hard) value lying in this interval. A cut norm interval that is far from zero represents less uniform and independent sampling. This figure shows the cut norm estimates for all four examined algorithms: WaRSwap, CoMoFinder, DIA-MCIS, and FANMOD. The cut norm estimates for DIA-MCIS are absent from C and D because this algorithm is not able to perform on large graphs with more that 2,035 nodes.

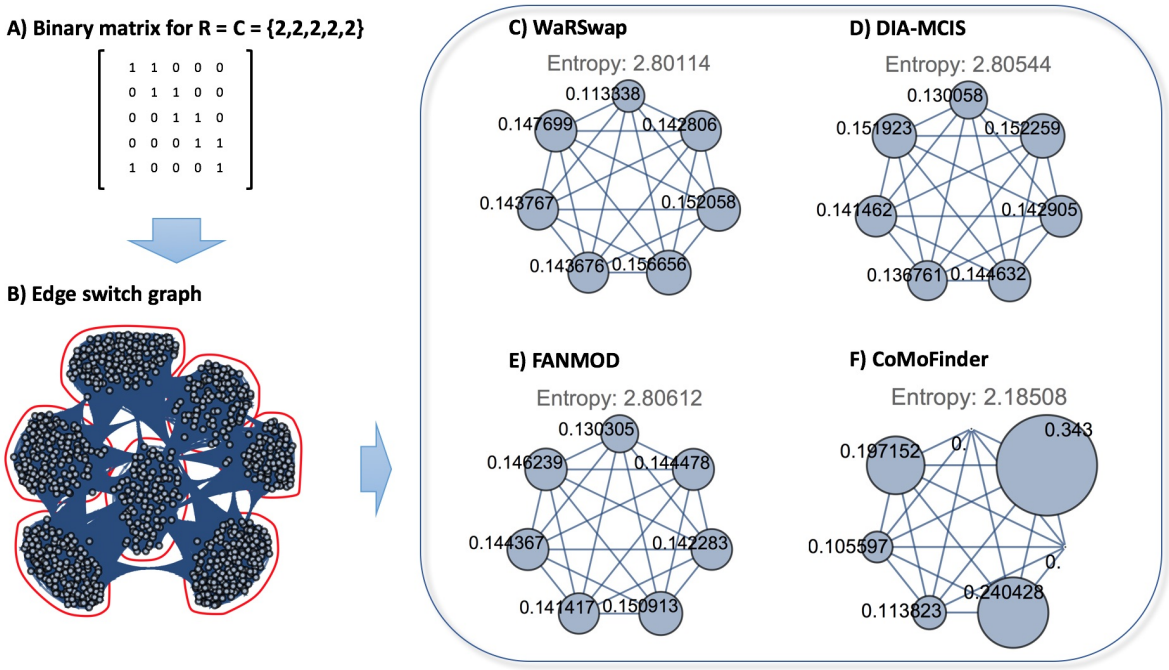

Figure S11: The ESG graph and cluster-time diagrams for an example even graph. A) The zero-one matrix representation of an even graph with degree sequence of R=C={2,2,2,2,2}. B) The ESG graph corresponding to the graph in part A. Running the graph clustering algorithm on the ESG graph detects seven different clusters. C-F) The cluster-time diagrams for each examined algorithm were computed and visualized.

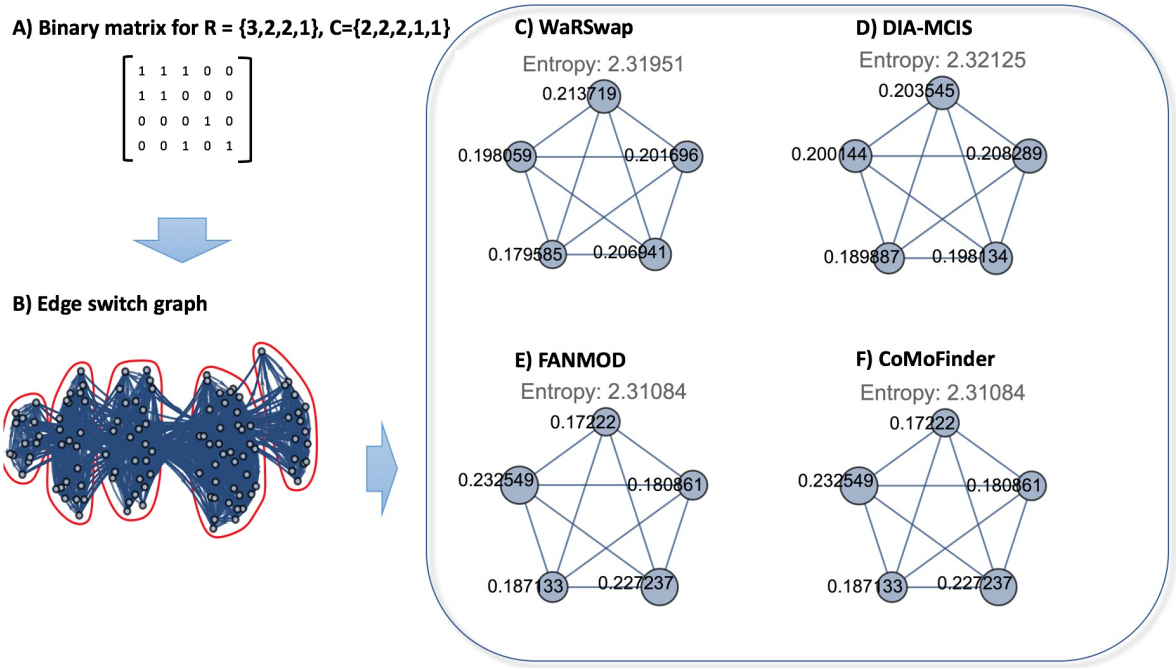

Figure S12: The ESG graph and cluster-time diagrams for an example hybrid graph. A) The zero-one matrix representation of an uneven graph with degree sequence of R={3,2,2,1}, C={2,2,2,1,1}. B) The ESG graph corresponding to the graph in part A. Running the graph clustering algorithm on the ESG graph detects five different clusters. C-F) The cluster-time diagrams for each examined algorithm were computed and visualized.

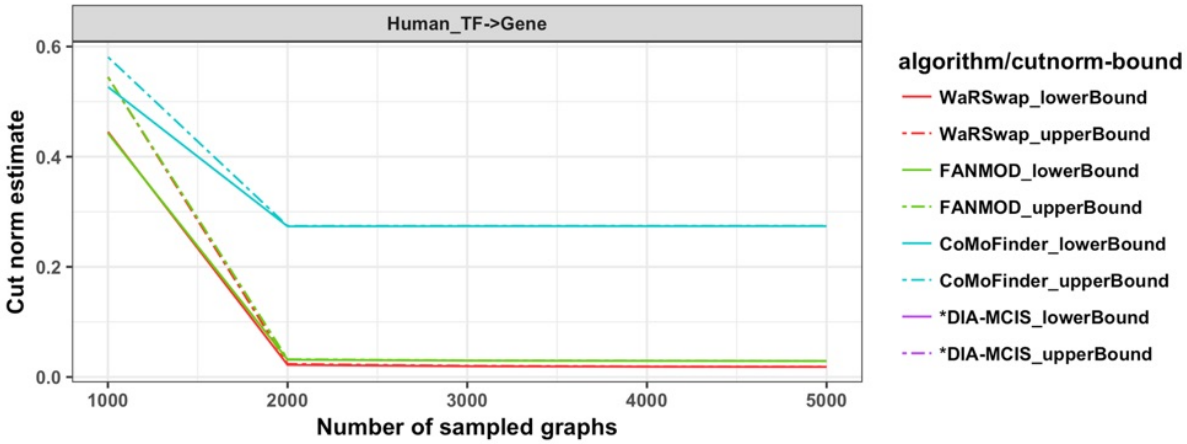

Figure S13: Relationship between the number of samples vs. sampling performance for Human TFGene network. All 5000 samples previously generated by each algorithm for the Human TFGene network were collected and subsampled into five sets (1000, 2000 ..., 5000 samples in each set, respectively). IndeCut was used to compute the cut norm estimates (lower and upper bounds) for each set of samples and algorithms. Cut norm values closer to zero represent a more uniform/independent sampling. This network has 9,055 nodes and 25,748 edges. *The cut norm estimates for DIA-MCIS are absent because this algorithm is not able to operate on networks with more than 2,035 nodes.

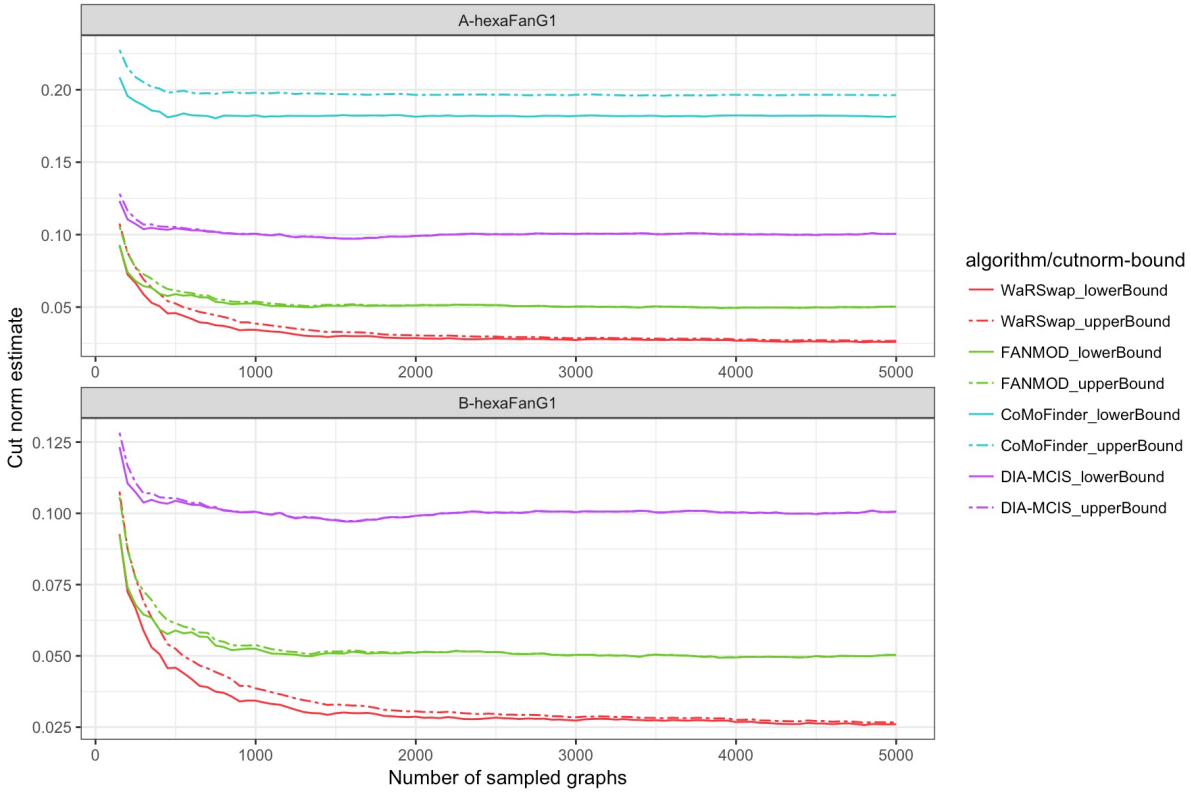

Figure S14: Sampling performance vs. the number of samples for graph hexaFanG1. All 5000 samples previously generated by each algorithm for hexaFanG1 were collected and subsampled into 25 sets (200, 400, 600, ..., 5000 samples in each set, respectively). IndeCut was used to compute the cut norm estimates (lower and upper bounds) for each set of samples and algorithms. Cut norm values closer to zero represent a more uniform/independent sampling. A) The relationship between the sampling performance and number of samples for all four examined algorithms is shown. B) The relationship between the sampling performance and number of samples for three algorithms WaRSwap, FANMOD, and DIA-MCIS is shown. The cut norm estimates for CoMoFinder were removed from this figure for ease of comparison (CoMoFinder has much larger cut norm estimates as compared to other three algorithms).

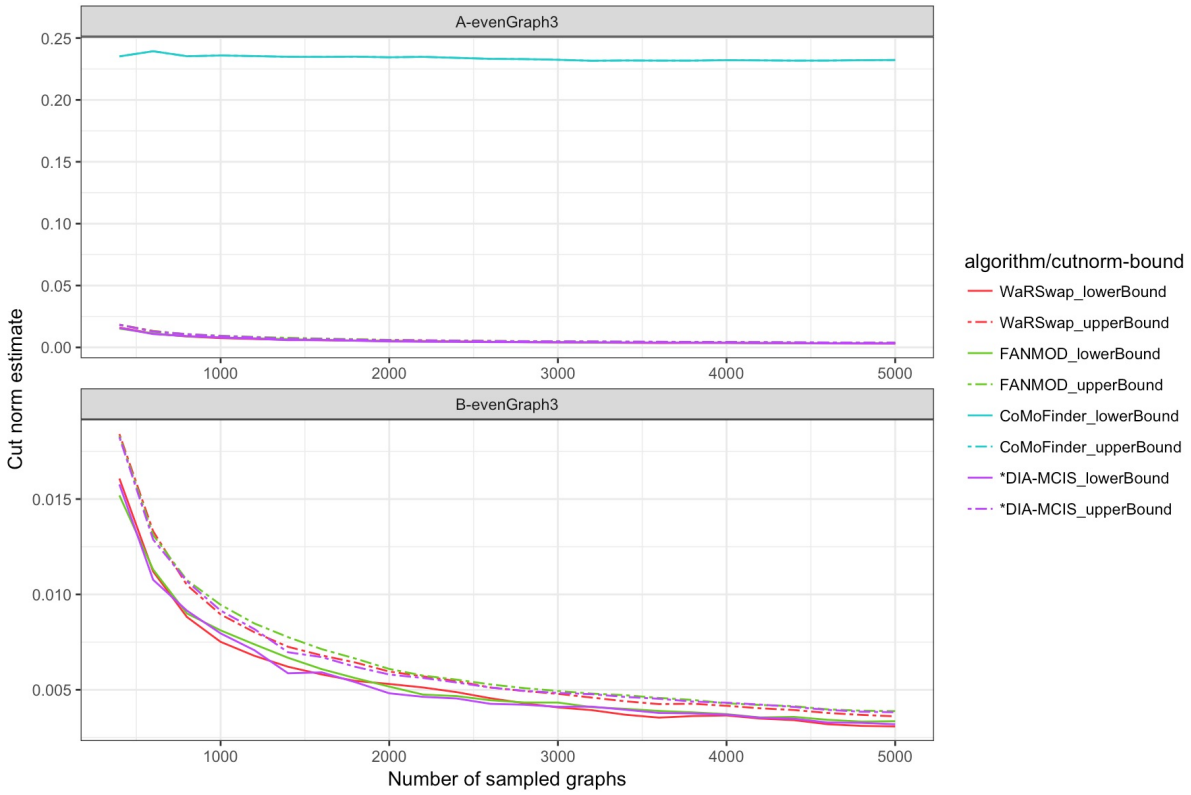

Figure S15: The cut norm estimates vs. the number of samples for graph evenGraph3. All 5000 samples previously generated by each algorithm for evenGraph3 network were collected and subsampled into 25 sets (200, 400, 600, ..., 5000 samples in each set, respectively). IndeCut was used to compute the cut norm estimates (lower and upper bounds) for each set of samples and algorithms. Cut norm values closer to zero represent a more uniform/independent sampling. A) The relationship between the sampling performance and number of samples for all four examined algorithms is shown. B) The relationship between the sampling performance and number of samples for three algorithms WaRSwap, FANMOD, and DIA-MCIS is shown. The cut norm estimates for CoMoFinder were removed from this figure for ease of comparison (CoMoFinder has much larger cut norm estimates as compared to other three algorithms).

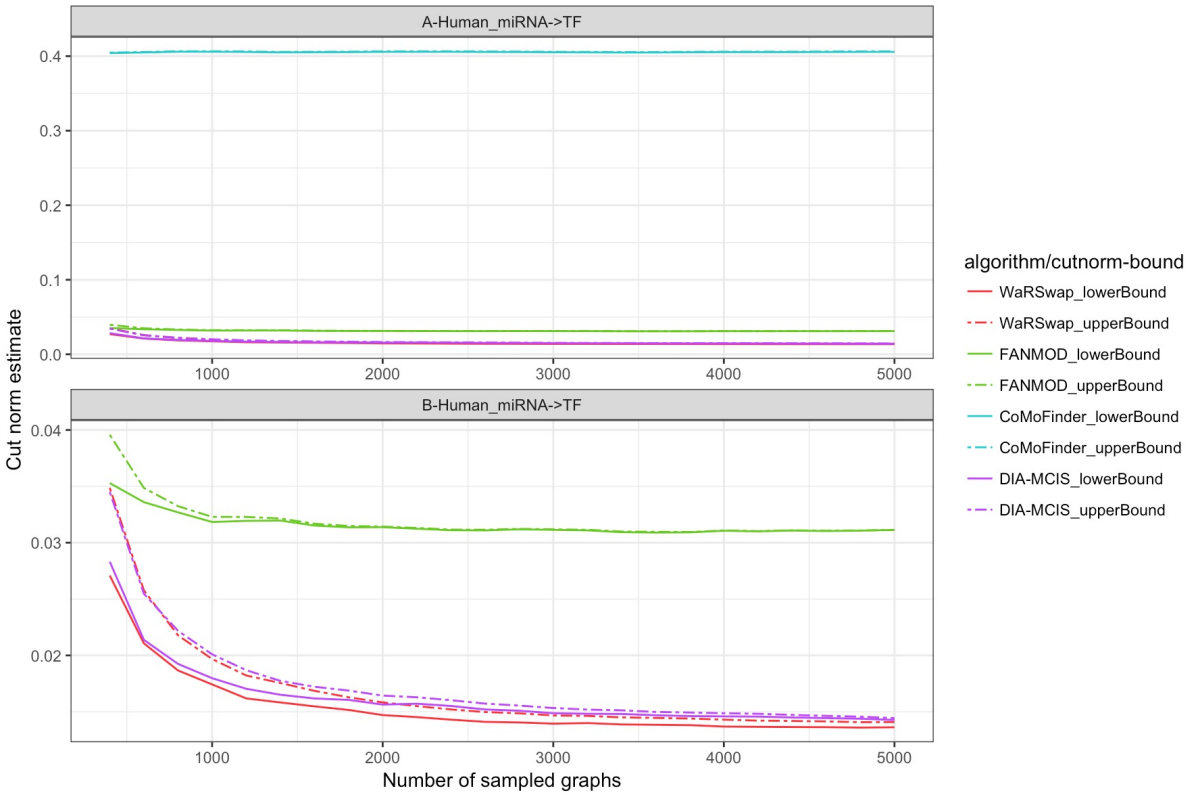

Figure S16: The cut norm estimates vs. the number of samples for Human miRNA→TF network. All 5000 samples previously generated by each algorithm for Human miRNA→TF network were collected and subsampled into 25 sets (200, 400, 600,. . . , 5000 samples in each set, respectively). IndeCut was used to compute the cut norm estimates (lower and upper bounds) for each set of samples and algorithms. Cut norm values closer to zero represent a more uniform/independent sampling. A) The relationship between the sampling performance and number of samples for all four examined algorithms is shown. B) The relationship between the sampling performance and number of samples for three algorithms WaRSwap, FANMOD, and DIA-MCIS is shown. The cut norm estimates for CoMoFinder were removed from this figure for ease of comparison (CoMoFinder has much larger cut norm estimates as compared to other three algorithms).

Table S1: Table of cut norm estimates for all examined graphs. The cut norm estimates closer to zero represents more uniform and independent sampling.

graphName	no_samples	Z_low cutnorm	Z_up cutnorm	WR_low cutnorm	WR_up cutnorm	FN_low cutnorm	FN_up cutnorm	comoF_low cutnorm	comoF_up cutnorm	diamcis_low cutnorm	diamcis_up cutnorm
uniFanG1	5000	5.000004	5.000004	0.0065	0.0066	0.0343	0.0343	0.015	0.0164	0.0057	0.0064
biFanG1	5000	10.000006	10.000006	0.0111	0.0122	0.0372	0.0373	0.8042	0.8068	0.1549	0.1549
triFanG1	5000	14.999994	14.999994	0.0198	0.0201	0.0527	0.0528	0.367	0.4123	0.1525	0.1525
tetraFanG1	5000	20.000003	20.000003	0.0241	0.0248	0.0582	0.0583	0.2778	0.2842	0.1358	0.1358
pentaFanG1	5000	24.99999	24.99999	0.0259	0.0264	0.0552	0.0552	0.2185	0.2315	0.1182	0.1182
F-hexaFanG1	5000	29.99998	29.99998	0.0261	0.0266	0.0498	0.0499	0.1814	0.1962	0.1011	0.1011
evenGraph1	5000	200.000013	200.000013	0.0034	0.0038	0.0032	0.0037	0.0677	0.0802	0.0033	0.0038
evenGraph2	5000	264.499966	264.5	0.0029	0.0036	0.0029	0.0035	0.2276	0.2276	0.003	0.0036
evenGraph3	5000	242.000044	242.000044	0.0029	0.0034	0.0031	0.0037	0.2323	0.2323	0.0031	0.0037
evenGraph4	5000	5.5	5.5	0.0132	0.0146	0.0127	0.0139	0.2123	0.2177	0.0119	0.0134
evenGraph5	5000	21.499986	21.499987	0.012	0.0138	0.0117	0.0136	0.1123	0.1137	0.011	0.0129
evenGraph6	5000	7.999996	7.999996	0.0211	0.0213	0.0283	0.0286	0.2373	0.2421	0.0259	0.0266
hybridGraph1	5000	9.499999	9.499999	0.0147	0.0161	0.0218	0.0221	0.3892	0.3892	0.0126	0.0139
hybridGraph2	5000	96.750016	96.750016	0.0199	0.0205	0.0399	0.0399	0.4097	0.4163	0.0277	0.0278
hybridGraph3	5000	111.000018	111.000018	0.0155	0.016	0.0272	0.0272	0.3184	0.3197	0.0232	0.0237
hybridGraph4	5000	42.499996	42.499996	0.0145	0.0157	0.0276	0.0276	0.3555	0.378	0.0204	0.0222
hybridGraph5	5000	20.000008	20.000008	0.0301	0.0306	0.0463	0.0464	0.36	0.3757	0.0198	0.0209
ecoli_TF-GENE	5000	97.499958	97.498955	0.0145	0.0169	0.0384	0.0388	0.344	0.3441	0.0142	0.0164
ecoli_TF-TF	5000	32.249996	32.250064	0.0144	0.0166	0.0244	0.025	0.26	0.26	0.0143	0.0163
Human_TF→TF	5000	161.000002	161.000002	0.0301	0.0301	0.0385	0.0385	0.5313	0.5318	0.0123	0.0129
Human_TF→miRNA	5000	309.250145	309.250145	0.0297	0.0312	0.0312	0.0312	0.1236	0.125	0.0248	0.0258
Human_TF→GENE	5000	6437.000388	6437.000388	0.0181	0.0183	0.029	0.029	0.274	0.275		
Human_miRNA→GENE	5000	28855.25551	28855.25551	0.0087	0.0088	0.0202	0.0202	0.2852	0.2853		
Human_miRNA→TF	5000	648.50001	648.50001	0.0136	0.014	0.031	0.031	0.4057	0.4062	0.0141	0.0143

Table S2: Runtime of IndeCut on all examined graphs. IndeCut evaluates graphs on the order of several thousand nodes and tens of thousands of edges within a few minutes to a few days using standard hardware. This table provides IndeCut's observed run time on each graph and algorithm. The miRNA→Gene layer in the human network allows us to provide run time given an extreme example with approximately 100,000 edges. To put these run times into perspective, network motif tools typically take several days simply to provide an output for graphs of this size, using a small number of iterations that does not guarantee meaningfully accurate performance (we discuss the number of iterations necessary for optimal performance for each sampling method in the next section). Using a commercial optimization package such as Guorbi or Mosek (in contrast to the open-source package CSDP that we use here) will result in speed improvements to IndeCut. Thus, considering time costs of running network motif finding algorithms themselves as well as the enormous potential laboratory costs of attempting to validate inaccurate results, IndeCut presents a very practical method for making an informed network motif discovery algorithm choice on biological networks of study.

Graph	Number of nodes	Number of edges	IndeCut run-time
uniFanG1	11	20	10 s
biFanG1	22	40	10 s
triFanG1	33	60	15 s
tetraFanG1	44	80	20 s
pentaFanG1	55	100	35 s
hexaFanG1	66	120	48 s
regularSmallG1	23	22	9 s
regularSmallG2	62	86	21 s
regularSmallG3	31	32	10 s
regularG1	80	800	27 s
regularG2	92	1058	30 s
regularG3	88	968	28 s
Human_TF→TF	174	644	52 s
Human_TF→miR	332	1237	2 min
Human_miR→TF	606	2594	5 min
Human_TF→GENE	9055	25748	4 days
Human_miR→GENE	12185	115421	14 days
Ecoli_TF→TF	140	129	2 min
Ecoli_TF→GENE	365	390	4 min

References

[1] Barvinok, A. (2010) On the number of matrices and a random matrix with prescribed row and column sums and 0–1 entries. *Advances in Mathematics,* **224**(1), 316–339.

[2] Alon, N. and Naor, A. (2006) Approximating the cut-norm via Grothendieck's inequality. *SIAM Journal on Computing,* **35**(4), 787–803.

[3] Borchers, B. (1999) CSDP, AC library for semidefinite programming. *Optimization methods and Software,* **11**(1-4), 613–623.

[4] Janson, S., Graphons, cut norm and distance, couplings and rearrangements. Technical report, Department of Mathematics, Uppsala University (2010).

[5] Wernicke, S. and Rasche, F. (2006) FANMOD: a tool for fast network motif detection. *Bioinformatics,* **22**(9), 1152–1153.

[6] Fusco, D., Bassetti, B., Jona, P., and Lagomarsino, M. C. (2007) DIA-MCIS: an importance sampling network randomizer for network motif discovery and other topological observables in transcription networks. *Bioinformatics,* **23**(24), 3388–3390.

[7] Megraw, M., Mukherjee, S., and Ohler, U. (2013) Sustained-input switches for transcription factors and microRNAs are central building blocks of eukaryotic gene circuits. *Genome biology,* **14**(8), 1.

[8] Liang, C., Li, Y., Luo, J., and Zhang, Z. (2015) A novel motif-discovery algorithm to identify co-regulatory motifs in large transcription factor and microRNA co-regulatory networks in human. *Bioinformatics,* **31**(14), 2348–2355.

[9] Chen, Y., Diaconis, P., Holmes, S. P., and Liu, J. S. (2005) Sequential Monte Carlo methods for statistical analysis of tables. *Journal of the American Statistical Association,* **100**(469), 109–120.

[10] Shen-Orr, S. S., Milo, R., Mangan, S., and Alon, U. (2002) Network motifs in the transcriptional regulation network of Escherichia coli. *Nature genetics,* **31**(1), 64–68.

[11] Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. (2008) On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering,* **20**(2), 172–188.