IndeCut evaluates performance of network motif discovery algorithms

Mitra Ansariola\textsuperscript{1,2}, Molly Megraw\textsuperscript{1,2,3,*} and David Koslicki\textsuperscript{1,4,*}

December 2, 2017

\textsuperscript{1} Center for Genome Research and Biocomputing, \textsuperscript{2} Department of Botany and Plant Pathology, \textsuperscript{3} Department of Computer Science, \textsuperscript{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 \( C \):

\[
\left| \Sigma(R, C) \right| \geq \left( \frac{mn}{mn} \right)! \left( \frac{m}{mn} \right)^m \left( \frac{n}{mn} \right)^n \prod_{i=1}^{m} \left( 1 + x_i y_j \right) \alpha(R, C).
\]

Taking the logarithm of \( F(x, y) \) 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 \( x^* \) and \( y^* \) be the vectors that obtain optimality in the definition of \( \alpha(R, C) \). Define \( Z \in \mathbb{R}^{m \times n} \) as

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

The cut norm is needed to state the next theorem.

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

\[
\| A \|_C = \max_{\substack{I \subseteq \{1, \ldots, n\} \atop J \subseteq \{1, \ldots, m\}}} \left| \sum_{i \in I, j \in J} A_{i,j} \right|. \tag{Cut Norm}
\]

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

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

Note that \( \| A \|_C = \max_S |\sigma_S(A)| \).

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 $|\sigma_S(Z - A(N))| = ||Z - A(N)||_C$. 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

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

Proof. Assuming that

$$(1 - \epsilon)\sigma_S(Z) \leq \sigma_S(A(N)) \leq (1 + \epsilon)\sigma_S(Z),$$

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

$$(1 - \epsilon)||Z||_C \leq \sigma_S(A(N)) \leq (1 + \epsilon)||Z||_C. \quad (3)$$

By hypothesis, $|\sigma_S(Z - A(N))| = ||Z - A(N)||_C$ and so along with linearity of $\sigma_S(\cdot)$, equation (3) implies that

$$||A(N) - Z||_C \leq \epsilon ||Z||_C.$$ 

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

$$\mathbb{P}\left\{ \frac{\frac{1}{N} \sum_{i=1}^{N} A_i - Z}{||Z||_C} \leq \epsilon \right\} \geq 1 - 2n^{-\kappa n}. \quad \Box$$

Given appropriate $R$, $C$, $\kappa$, $\delta$, and $\epsilon$, the contrapositive of this result implies that if $\frac{||A(N) - Z||_C}{||Z||_C}$ 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

$$\frac{||A(N) - Z||_C}{||Z||_C}$$

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 $|| \cdot ||_C$ 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 ($|| \cdot ||_{\infty \rightarrow 1}$) that can be approximated with a semidefinite relaxation. We then round the solution of the semidefinite relaxation to get an estimate of $|| \cdot ||_{\infty \rightarrow 1}$ and hence of $|| \cdot ||_C$. We begin with definitions of the norms of interest.
**Definition 3.** Let $A \in \mathbb{R}^{m \times n}$. Define the following norms by

$$
\|A\|_{\infty \to 1} = \maximize_{x_i \in \{-1,+1\}, y_j \in \{-1,+1\}} \sum_{i,j} A_{i,j} x_i y_j \quad (\infty \to 1 \text{ Norm})
$$

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

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

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

$$
\|A\|_{\text{SDR}} = \frac{1}{2} \maximize_X \text{tr}(CX)
$$

subject to $\text{tr}(F_k X) = a_k, k = 1, \ldots, m + n$

$$
X \succeq 0,
$$

for

$$
C = \begin{bmatrix} 0 & A \\ A & 0 \end{bmatrix}, \quad F_k = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{o.w.} \end{cases}
$$

and $a_k = 1$ for $k = 1, \ldots, m + n$.

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

**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 $\|A\|_{\infty \to 1} = 4\|A\|_C$.

**Proof.** For $I \subseteq \{1, \ldots, n\}$ and $J \subseteq \{1, \ldots, m\}$ the sets achieving the maximum in the definition of $\|A\|_C$, define $x_i = 1$ for $i \in I$, $x_i = -1$ for $i \not\in I$ and $y_j = 1$ for $j \in J$, $y_j = -1$ for $j \not\in J$. Then

$$
\|A\|_C = \sum_{i,j} A_{i,j} \frac{1 + x_i y_j}{2}
$$

$$
= \frac{1}{4} \left( \sum_{i,j} A_{i,j} + \sum_{i} A_{i,j} x_i + \sum_{j} A_{i,j} y_j + \sum_{i,j} A_{i,j} x_i y_j \right)
$$

$$
= \frac{1}{4} \sum_{i,j} A_{i,j} x_i y_j
$$

$$
= \frac{1}{4} \|A\|_{\infty \to 1}.
$$

**Cut norm estimates**

In [2] Section 5.1, an algorithm was presented that computes bounds on $\|\cdot\|_{\infty \to 1}$. 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 $i = 1, \ldots, m, j = 1, \ldots, n$ be the optimal vectors obtained from the computation of $\|A\|_{\text{SDR}}$. Let $g_i \sim N(0, 1)$. $i = 1, \ldots, m + n$, be independent standard normal random variables and let $G = (g_1, \ldots, g_{m+n})$. Let $x_i = \text{sign}(u_i \cdot G)$ and $y_j = \text{sign}(v_j \cdot G)$.
Now, \( \sum_{i,j} A_{i,j} x_i y_j \leq \|A\|_{\infty \to 1} \) since \( \|A\|_{\infty \to 1} \) is the maximum value. However, there is a positive probability that \( \sum_{i,j} A_{i,j} x_i y_j = \|A\|_{\infty \to 1} \). To observe this fact, let \( x^*_i, y^*_j \in \{-1, +1\} \) be such that \( \|A\|_{\infty \to 1} = \sum_{i,j} A_{i,j} x^*_i y^*_j \). 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 \( 2^{-m-k} \), 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 \( \|A\|_{\infty \to 1}^{\text{est}} \). As a result, \( \|A\|_{\infty \to 1}^{\text{est}} \leq \|A\|_{\infty \to 1} \leq \|A\|_{\text{SDR}} \), so combining these with proposition \([4]\) we have the following estimation of the cut norm:

\[
\frac{1}{4} \|A\|_{\infty \to 1}^{\text{est}} \leq \|A\|_C \leq \frac{1}{4} \|A\|_{\text{SDR}}.
\]

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

\[
\frac{\|A(N) - Z\|_C}{4\|Z\|_C}.
\]

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 \( A \) and \( B \). If the upper bound for one algorithm (say, \( A \)) falls below the lower bound of the other algorithm (say, \( B \)), then we can be sure that the cut norm quantity of interest for \( A \) is smaller than for \( B \). As a consequence of the previous section, this implies that we can be more confident that \( B \) 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

\[
\|A(N) - Z\|_{\text{SDR}} < \|B(N) - Z\|_{\infty \to 1}^{\text{est}}
\]

then equation \([5]\) implies that

\[
\frac{\|A(N) - Z\|_C}{4\|Z\|_C} < \frac{\|B(N) - Z\|_C}{4\|Z\|_C}.
\]

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

\[
\left( \frac{\|Z - A(N)\|_{\infty \to 1}^{\text{est}}}{4\|Z\|_C}, \frac{\|Z - A(N)\|_{\text{SDR}}}{4\|Z\|_C} \right) \cap \left( \frac{\|Z - B(N)\|_{\infty \to 1}^{\text{est}}}{4\|Z\|_C}, \frac{\|Z - B(N)\|_{\text{SDR}}}{4\|Z\|_C} \right) \neq \emptyset,
\]

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

\[
\text{IndeCut}(Z,A,N) = \left( \frac{\|Z - A(N)\|_{\infty \to 1}^{\text{est}}}{4\|Z\|_C}, \frac{\|Z - A(N)\|_{\text{SDR}}}{4\|Z\|_C} \right)
\]

to compare uniformity/independence of motif finding algorithms.
Algorithm 1: Cut norm lower bound

Input:
\[ A \in \mathbb{R}^{m \times n} \]
\[ c \in \mathbb{N} \]
\[ u_i, v_j \in \mathbb{R}^{m+n} \]

Initialization:
\[ its = 0 \]
\[ bound = 0 \]

Iterations:
\[ \text{while } its < c \text{ do} \]
\[ G = (g_1, \ldots, g_{m+n}) \]
\[ \text{for } i = 1, \ldots, m \text{ do} \]
\[ x_i = \text{sign}(u_i \cdot G) \]
\[ \text{end for} \]
\[ \text{for } j = 1, \ldots, n \text{ do} \]
\[ y_j = \text{sign}(v_j \cdot G) \]
\[ \text{end for} \]
\[ \text{temp} = \sum_{i,j} A_{i,j} x_i, y_j \]
\[ \text{if } \text{temp} > \text{bound} \text{ then} \]
\[ \text{bound} = \text{temp} \]
\[ \text{end if} \]
\[ its = its + 1 \]
\[ \text{end while} \]

Output:
\[ ||A||_{\infty \rightarrow 1} = \text{bound} \]

(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 → TF, TF → gene, miRNA TF, TF TF, TF → miRNA, TF → TF, TF → 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 \(M_{m \times n}\) with \(m\) rows and \(n\) columns where \(M_{ij}\) is 1 if node \(i\) has a directed link to node \(j\). 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 \(\text{deg}_0\), and assigns \(\text{deg}_0\) is 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 → TF, TF → miRNA, TF → gene, miRNA → TF, and miRNA → 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/](http://megraw.cgrb.oregonstate.edu/software/WaRSwapSoftwareApplication/) and R implementation from [http://megraw.cgrb.oregonstate.edu/software/WaRSwap](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 → 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: \texttt{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 \cite{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 \texttt{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 \cite{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 \cite{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 $A$, the output of $A$ can be viewed as sampling vertices of the ESG. Define a count vector $count^A \in \mathbb{N}^L$ as a vector indexed by the clusters found above, with $count^A_i$ being equal to the number of times the algorithm $A$ returned a graph found in cluster $i$. 

8
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 $\text{count}^A_i$. The entropy of the vector $\text{count}^A$ is also calculated to quantify how equally (or unequally) the algorithm $A$ samples graphs belonging to each cluster: $-\sum_{i=1}^{L} \frac{\text{count}^A_i}{\sum_j \text{count}^A_j} \log \left( \frac{\text{count}^A_i}{\sum_j \text{count}^A_j} \right)$. Larger entropy values indicate that the algorithm $A$ 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 \textit{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 than 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}. 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 \( R = \{3,2,2,1,1\} \), \( C = \{2,2,2,1,1\} \). 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.

<table>
<thead>
<tr>
<th>graphName</th>
<th>no_samples</th>
<th>lowcutnorm</th>
<th>upcutnorm</th>
<th>WR_lowcutnorm</th>
<th>WR_upcutnorm</th>
<th>comoF_lowcutnorm</th>
<th>comoF_upcutnorm</th>
<th>diamcis_lowcutnorm</th>
<th>diamcis_upcutnorm</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniFanG1</td>
<td>5000</td>
<td>5.000004</td>
<td>5.000004</td>
<td>0.0065</td>
<td>0.0066</td>
<td>0.0434</td>
<td>0.0434</td>
<td>0.0156</td>
<td>0.0156</td>
</tr>
<tr>
<td>biFanG1</td>
<td>5000</td>
<td>10.000006</td>
<td>10.000006</td>
<td>0.0111</td>
<td>0.0112</td>
<td>0.0418</td>
<td>0.0404</td>
<td>0.0149</td>
<td>0.0149</td>
</tr>
<tr>
<td>triFanG1</td>
<td>5000</td>
<td>14.999994</td>
<td>14.999994</td>
<td>0.0198</td>
<td>0.0203</td>
<td>0.0427</td>
<td>0.0428</td>
<td>0.0213</td>
<td>0.0215</td>
</tr>
<tr>
<td>tetraFanG1</td>
<td>5000</td>
<td>20.000004</td>
<td>20.000004</td>
<td>0.0241</td>
<td>0.0248</td>
<td>0.0452</td>
<td>0.0463</td>
<td>0.0278</td>
<td>0.0284</td>
</tr>
<tr>
<td>pentaFanG1</td>
<td>5000</td>
<td>24.999999</td>
<td>24.999999</td>
<td>0.0259</td>
<td>0.0264</td>
<td>0.0462</td>
<td>0.0473</td>
<td>0.0294</td>
<td>0.0311</td>
</tr>
<tr>
<td>F-hexaFanG1</td>
<td>5000</td>
<td>29.99998</td>
<td>29.99998</td>
<td>0.0284</td>
<td>0.0286</td>
<td>0.0498</td>
<td>0.0499</td>
<td>0.0314</td>
<td>0.0316</td>
</tr>
<tr>
<td>evenGraph1</td>
<td>5000</td>
<td>200.000013</td>
<td>200.000013</td>
<td>0.0034</td>
<td>0.0038</td>
<td>0.0032</td>
<td>0.0037</td>
<td>0.0033</td>
<td>0.0033</td>
</tr>
<tr>
<td>evenGraph2</td>
<td>5000</td>
<td>264.499966</td>
<td>264.5</td>
<td>0.0029</td>
<td>0.0036</td>
<td>0.0037</td>
<td>0.0037</td>
<td>0.0037</td>
<td>0.0038</td>
</tr>
<tr>
<td>evenGraph3</td>
<td>5000</td>
<td>242.000044</td>
<td>242.000044</td>
<td>0.0029</td>
<td>0.0034</td>
<td>0.0034</td>
<td>0.0035</td>
<td>0.0035</td>
<td>0.0036</td>
</tr>
<tr>
<td>evenGraph4</td>
<td>5000</td>
<td>5.5</td>
<td>5.5</td>
<td>0.0132</td>
<td>0.0146</td>
<td>0.0127</td>
<td>0.0139</td>
<td>0.0124</td>
<td>0.0134</td>
</tr>
<tr>
<td>hybridGraph1</td>
<td>5000</td>
<td>21.499986</td>
<td>21.499987</td>
<td>0.0142</td>
<td>0.0138</td>
<td>0.0117</td>
<td>0.0146</td>
<td>0.0112</td>
<td>0.0117</td>
</tr>
<tr>
<td>hybridGraph2</td>
<td>5000</td>
<td>31.999956</td>
<td>31.999956</td>
<td>0.0121</td>
<td>0.0123</td>
<td>0.0128</td>
<td>0.0129</td>
<td>0.0122</td>
<td>0.0126</td>
</tr>
<tr>
<td>hybridGraph3</td>
<td>5000</td>
<td>4.499959</td>
<td>4.499959</td>
<td>0.0117</td>
<td>0.0116</td>
<td>0.0121</td>
<td>0.0122</td>
<td>0.0092</td>
<td>0.0092</td>
</tr>
<tr>
<td>hybridGraph4</td>
<td>5000</td>
<td>96.750016</td>
<td>96.750016</td>
<td>0.0039</td>
<td>0.0039</td>
<td>0.0039</td>
<td>0.0039</td>
<td>0.0039</td>
<td>0.0039</td>
</tr>
<tr>
<td>hybridGraph5</td>
<td>5000</td>
<td>20.000008</td>
<td>20.000008</td>
<td>0.0030</td>
<td>0.0030</td>
<td>0.0046</td>
<td>0.0064</td>
<td>0.0036</td>
<td>0.0036</td>
</tr>
<tr>
<td>Human_TF→GENE</td>
<td>5000</td>
<td>161.000002</td>
<td>161.000002</td>
<td>0.0081</td>
<td>0.0081</td>
<td>0.0085</td>
<td>0.0085</td>
<td>0.0034</td>
<td>0.0034</td>
</tr>
<tr>
<td>Human_TF→miRNA</td>
<td>5000</td>
<td>303.250145</td>
<td>303.250145</td>
<td>0.0029</td>
<td>0.0032</td>
<td>0.0032</td>
<td>0.0032</td>
<td>0.0123</td>
<td>0.0123</td>
</tr>
<tr>
<td>Human_TF→TF</td>
<td>5000</td>
<td>543.000038</td>
<td>543.000038</td>
<td>0.0081</td>
<td>0.0081</td>
<td>0.0082</td>
<td>0.0082</td>
<td>0.0082</td>
<td>0.0082</td>
</tr>
<tr>
<td>Human_miRNA→TF</td>
<td>5000</td>
<td>3058.2551</td>
<td>2885.2554</td>
<td>0.0087</td>
<td>0.0088</td>
<td>0.0082</td>
<td>0.0082</td>
<td>0.0082</td>
<td>0.0082</td>
</tr>
<tr>
<td>Human miRNA→GENE</td>
<td>5000</td>
<td>168.5600</td>
<td>168.5600</td>
<td>0.0116</td>
<td>0.0114</td>
<td>0.0104</td>
<td>0.0104</td>
<td>0.0104</td>
<td>0.0104</td>
</tr>
</tbody>
</table>
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.

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


