ISCI, Volume *19*

Supplemental Information

An Algorithmic Information Calculus for Causal Discovery and Reprogramming Systems

Hector Zenil, Narsis A. Kiani, Francesco Marabita, Yue Deng, Szabolcs Elias, Angelika Schmidt, Gordon Ball, and Jesper Tegnér

Supplementary Information

Section 1: Glossary of Terms, Concepts and Definitions

Algorithmic causality: We define as the causal index c of a dynamical system S_t running over time t as the smallest c such that $C(S_t) = |S_t| - c$, where | X | denotes the size of X. The difference $|S_t| - C(S_t)$ is thus an approximation of the causality of S_t. The causal content c of a non-causal system approximates log t, i.e. is very small, meaning that $C(S_t) \sim |S_t|$, and the trajectory of S_t is algorithmically random. For causal systems, we have it that $C(S_t) - C(S_{t+1}) \sim log_2$ t, i.e. the complexity of a causal system S is driven by its evolution time t. All logarithms are in base two if not otherwise indicated.

Algorithmic perturbation analysis: Is the estimation of the effects of perturbations (e.g. by removal/knockout) of an element e (or set of elements) from S, denoted by S\e, on the original algorithmic information content C(S). Without loss of generalization, let's take as a system s, a network G = {V(G), E(G)}, a dynamic system, with V(G) a set of nodes and E(G) a set of links connecting nodes in V(G).

Negative information element (e.g. a node or edge): an element (or set) e in G such that: $C(G) - C(G\backslash e) < - log_2|V(G)|$, i.e. the removal of e moves G towards randomness.

Positive information element (e.g. a node or edge): an element (or set) e in G such that: $C(G) - C(G\backslash e) > log_2|V(G)|$, i.e. the removal of e moves G away from randomness.

Neutral information element (e.g. a node or edge): an element (or set) e in G such that e is neither positive or negative:
$- log_2|V(G)| \leq C(G) - C(G\backslash e) \leq log_2|V(G)|$, where |V(G)| is the size of the system, e.g. the vertex count of a network G.

Algorithmic system inference of its generating mechanism: Let s be a dynamical system if $C(s_t) - C(s\backslash e) \sim log_2 t$ we then call e a neutral perturbation. A perturbation e thus does not change the generating mechanism of s and s_t can be recovered from $s_t\backslash e$ because $s_{t+1}\backslash e = s_t$. Otherwise, e is disruptive (positive or negative), with a degree of disruptiveness $C(s_t\backslash e) - C(s_t)$. In general, $C(s_t) - C(s_{t-n}) \sim n\ log_2 t$, providing the means to reverse a system in time and reveal its possible generating mechanism in the process. If the system is not reversible, several generating models may be formulated, thereby producing optimal hypotheses in the form of generative models.

***Spectra*(G)**: the list of all non-integer algorithmic-information contribution values of each element of G (e.g. edges or nodes, or both).

***Powerset spectra*(G)**: the list of all non-integer algorithmic-information values of each element in the powerset of elements of G (e.g. edges or nodes or both).

Red shifted *spectra*(G): *spectra*(G) that contain more elements whose removal moves G more towards than away from randomness.

Blue shifted *spectra*(G): *spectra*(G) that contain more elements whose removal moves G away from rather than towards randomness.

σ(G): the information signature (or just signature) of G is *spectra*(G) list sorted from largest to smallest value. (see Extended Data Figures. 1-2).

Δ(s): the instantaneous programmability value of an element s in σ(G), indicating how fast or slowly s can move G towards or away from randomness**.** Formally,
$\Delta(s) = |\sigma_i(G) - \sigma_i - 1(G) / P(\sigma_i(G)) - P(\sigma_i - 1(G))|$.

Incoherent information set: a set whose individual elements or subsets have different information contribution values than the whole set.

Coherent information set: a set whose individual elements or subsets have the same information contribution value as the whole set.

Information sensitivity: the derivative of the absolute max value of the programmability of a graph in the (re)programmability curve (see Figures S1-S4), but numerically calculated by the rate of change of σ(G\e) versus σ(G) for all elements (or sets) e in G, i.e. the list of signatures for all e (or signature of signatures of G) capturing the non-linear effects of perturbations on G.

MILS: Minimal Information Loss Sparsification is a method to identify neutral elements that have zero or negligible algorithmic-information content value in a system or network, and can thus safely be removed, ensuring minimal information loss.

MAR: a Maximal Algorithmic Random graph (or system) G is an Erdős -Rényi (E-R) graph that is algorithmically random, i.e. whose shortest possible computer description is not (much) shorter than |E(G)|, where |E(G)| is the number of edges of G; or, |E(G)| – C(G) < c.

1st Order randomness deficiency: The algorithmic-information distance between a network/system and its algorithmically randomised version, e.g. a MAR graph for networks.

2nd Order randomness deficiency: The difference between information signatures by, e.g., Kolmogorov-Smirnoff distance, i.e. how removed a network is from its algorithmic (non-causal) randomisation.

Simply directed graph: is the transformation of an undirected graph into a directed one such that the edge directions are chosen to minimise the number of independent paths and number of path collisions.

MAD: denotes the median absolute deviation, and is defined by:

$$MAD = median\ (|X_i - median(X)|).$$

MAD is a robust measure of the variability of a univariate sample.

Relative (re)programmability: $Pr(G) := MAD(\sigma(G))) / n$ or 0 if n = 0, where n = max(|σ(G)|). This index measures the shape of $\sigma_P(\mathrm{G})$ and how it deviates from other distributions (e.g. uniform or normal).

Absolute (re)programmability: $PA(G) := |S(\sigma P(G)) - S(\sigma N(G))| / m$, where m := $max(S(\sigma P(G)), S(\sigma N(G)))$, where $m = max(S(\sigma P(G)), S(\sigma N(G)))$ and S is an interpolation function. This measure of reprogrammability captures not only the shape of $\sigma_P(\mathrm{G})$ but also the sign of $\sigma_P(\mathrm{G})$ above and below x = 0.

Programmability landscape: the Cartesian product $Pr(G)\ x\ PA(G)$.

Combined (re)programmability: $||V_R(G)|| = \sqrt{P_R^2(G) - P_A^2(G)} \leq \sqrt{2}$.
The combined reprogrammability is a metric induced by the norm ||V_R (G)|| defined by the Euclidean distance between two (re)programmability indices. This metric combines the relative and absolute (re)programmability indices, and takes into equal account both the sign of the signature σ(G) of G and the shape of σ(G), consequently minimizing the impact of uncertain sign estimations due to (convergent) errors in the calculation of algorithmic complexity (Zenil and Kiani, 2016) attributable to boundary conditions (see Graph Algorithmic Probability as Upper Bounds to Graph Randomness).

Natural (re)programmability: is the expected theoretical (re)programmability of a system or network, compared to its estimated (re)programmability, e.g. for a complete graph all nodes and all edges should have the same algorithmic-information contribution, and thus σ(G) can be analytically derived (a flat uniform distribution $x = log\ |V(G)|$ with $|V(G)|$ the node count of G). Thus all the nodes of a complete graph are 'slightly' positive (or more precisely, neutral, if they are 'positive' by only $log\ |V(G)|$).

Algorithmic-information Causal Interventionist Calculus
The core of the causal calculus is based upon the change of complexity of a system subject to perturbations, particularly the direction (sign) and magnitude of the difference of algorithmic information content C between two graphs G and G', e.g. the removal of e from G (denoted by G\e). The difference | C(G) – C(G\e) | (see Supplement, Section 1) is an estimation of the shared algorithmic mutual information(Chaitin, 1987) of G and G\e. If e does not contribute to the description of G, then $|\ C(G) - C(G\backslash e)\ | \sim log_2|V(G)|$, where |V(G)| is the node count of G, i.e. the difference will be very small and at most a function of the graph size and thus C(G) and C(G\e) have almost the same complexity. If, however, $|\ C(G) - C(G\backslash e)\ | < log_2|V(G)|$ bits, then G and G\e share at least n bits of algorithmic information in element e, and the removal of e results in a loss of information. In contrast, if $C(G) - C(G\backslash e) > n$, then e cannot be explained by G alone nor is it algorithmically not contained/derived from G, and it is, therefore, a fundamental part of the description of G with e as a generative causal mechanism in G, or else it is not part of G but has to be explained independently, e.g. as noise. Whether it is noise or part of the generating mechanism of G depends on the relative magnitude of n with respect to C(G) and to the original causal content of G itself. If G is random then the effect of e will be small in either case, but if G is richly causal and has a very small generating program, then e as noise will have a greater impact on G than would removing e from the description of an already short description of G. However, if | C(G) – C(G\e) | $\leq log_2|V(G)|$, where |V(G)| is the vertex count of G, then e is contained in the algorithmic description of G and can be recovered from G itself (e.g. by running the program from a previous step until it produces G with e from G\e).

For example, in a complete graph K_{10} (Fig. 1a,b), the removal of any single node leads to a logarithmic reduction in its algorithmic complexity, but the removal of any single edge leads to

an increase of randomness. The former because the result is simply another complete graph of a smaller size, and the latter because the deleted link would need to be described after the description of the complete graph itself. However, the removal of node 1 (Fig. 1 b) is equivalent to the removal of the set of all edges connecting to node 1, so the set of all these edges is a positive information set, even though all its individual edges are negative, a nonlinear phenomenon that we call *information incoherence*. Connecting two complete graphs at a random node (Figure 1c) designates the connecting link as positive because its removal pushes the network towards simplicity, the minimal description of 2 K_{10} graphs being shorter than the minimal description of two K_{10} graphs plus the description of the missing link at random points. Such a link can also be seen as an element connecting 2 networks, hence a network of networks. Its identification and removal would thus reveal the separation between two networks. In general, positive elements will identify the major structures generated by the most likely (and simplest) generating mechanism given the observation, and odd elements will stand out as negative, thereby identifying layers of networks that are independent of separable generating mechanisms, even removing apparent noise (external information) from the signal (the system's natural evolution) when such networks are richly causal. Random graphs are node- and edge-*blueshifted* (see Fig. 1g and Supplementary Information Glossary Section 1); simple graphs such as complete or wheel graphs are edge-*redshifted*. Perturbing (e.g. knocking-out) a node and recalculating the spectra changes the original spectrum in what is clearly a non-reductionistic approach to characterizing networks. All the methods introduced here also work on directed (e.g. Fig. 1d) and weighted graphs without any loss of generality.

Real-world networks as generated by physical laws are recursive according to classical mechanics (deterministic and reversible) and are thus on the left side in the schematic Extended Data Figure. 1, but they may also contain information about other interacting systems or be captured in a transient state that incorporates external signals pushing the networks towards randomness. We have quantified this concept by proposing different (Re) Programmability indices (see Supplement Section 1). Extended Data Figure. One summarizes some of the theoretical expectations and numerical results. There is a thermodynamic argument as to why the curve is negatively skewed: while it is easy and fast to move regular networks towards randomness as a function of the number of edges—there being about $|E(G)|$ ways to move the network towards randomness such that the description of G moves to $|G| + |e|$, i.e. the description of, say, an edge removed, where $|E(G)|$ is the edge count of G—there are far fewer ways to move a random network away from randomness. A MAR graph, for example, cannot be moved by edge or node deletion more than $\log |E(G)|$. The result is compatible with the asymmetries in energy landscapes between moving systems towards fewer future attractors versus moving them back to states of a greater number of attractors, the latter requiring much more energy than the former.

Minimal Information Loss Sparsification (MILS)

Our causal algorithmic calculus defines an optimal parameter-free dimension reduction algorithm, which minimizes information loss while reducing the size of the original (network) object. The Minimal Information Loss Sparsification (or MILS) method is based on removing neutral elements while preserving the information content of a network, and therefore its properties, and it can be used for reduction by minimizing the loss of any informational feature of G that needs to be described and cannot be compressed into some shorter description of G (see Transparent Methods Supplement Section 2 for the pseudocode and evaluation).

Maximal Algorithmic Randomness Preferential Attachment (MARPA) algorithm

The Maximal Algorithmic Randomness Preferential Attachment (MARPA) algorithm (MARP) (see Supplement Section 2 for the pseudocode and evaluation) can be viewed as a reverse algorithm in comparison to MILS. MARPA seeks to maximize the information content of a graph G by adding new edges (or nodes) at every step. The process approximates a network of a given size that has the largest possible algorithmic randomness and is also an Erdős-Rényi (ER) graph. An approximation of a 'Maximal' Algorithmic- Random (MAR) graph can be produced as a reference object whose generating program is not smaller than the network (data) itself and can better serve in maximum (algorithmic-) entropy modelling. See Supplement Section 1 for the proof of the existence of ER graphs that are not maximal algorithmic-random graphs

Dynamical simulations using Boolean networks

A Boolean network consists of a discrete set of Boolean variables each of which has a Boolean function (here, always the same for each node), which takes inputs from a subset of these variables. We conducted the first experiment on single-node and single-edge deletion effects on all possible Boolean networks with up to size n = 5 nodes, and with XOR, AND, and OR as Boolean functions. The output of a Boolean network is the state of the numbered sequence of states of its nodes. In a Boolean model in which a network is represented by a set of n Boolean variables, either Off (0) or On (1), the number of attractors cannot exceed 2^{n2^n}.

In general, in a connected network, each node is controlled by a subset of other nodes. The size of the controlling subset for each network depends on the connectivity pattern in the network [3, 4]. For example, in an E-R random graph with edges equally distributed with edge density p, if we change the state of any arbitrary node in the initial state, the effect on the dynamics of a network should be about the same on average, and this means the basin of attraction remains mostly unchanged. If the basin of attraction is of size M, the number of attractors is $(2^n)/M$. The size of M will depend on the network density p with $M << 2^n$. However, in a simply connected complete graph (minimizing edge collision c.f. Sup. Inf. Glossary), all nodes control all other nodes, and there is only one attractor with the basin of attraction size 2^n. In modular scale-free networks, not all edges are statistically equally distributed, and only a few nodes control many others, unlike an E-R random network, and they have a significantly greater basin of attraction sizes and therefore a smaller number of attractors(Aldana, 2003a; Wuensche, 2004; Espanés, Osses and Rapaport, 2016).

We estimated the algorithmic-information contribution of every node n (and every edge e) over all possible 33,554,432 5-node graphs. The estimation of the algorithmic-information contribution (see Supplement Section 1)) considered all vertices in the same orbit of the automorphism group of G, Aut(G), and the min of the information value C(G\n) with respect to the largest component of G according to the unlabelled definition of algorithmic complexity for unlabelled graphs in [26 (main text)], thus correcting minor deviations of estimations of the complexity of C(G\n) by BDM due to boundary conditions[20 (main text)]. The calculation of C(G') for every G' in Aut(G) is, however, not feasible in general, as the production of Aut(G) and thus the calculation of C(G') for all G' in Aut(G) is believed to be in NP, thereby making the brute force exploration computationally intractable. However, it has also been shown that estimations of K(G) are similar to K(Aut(G))) up to a constant (the size of the graph generating program)[26 (main text)].

We performed the same edge perturbation experiments, removing all edges, one at a time, from larger graphs, [Fig3e] and comparing with state-of-the-art algorithms[5] the largest eigenvalue, number of different eigenvalues and number of attractors on the largest remaining connected component of the larger graphs. The experiment was repeated with Boolean functions AND, OR and XOR leading to the same results.

One can then apply uninformed perturbations to move networks towards statistical randomness based on this algorithmic-information calculus, and in a controlled fashion towards and away from algorithmic randomness, thus taking into account non-statistical and non-linear effects of the system as a generating mechanism, providing a sequence of causal interventions to move networks and systems at the level of the (hypothesized) generating model in order to reveal first principles and to control the side effects of such a system's manipulation at every step.

Random versus regular networks are sensitive in different ways. While an algorithmic- random network is hard to move fast along its algorithmic -random location (Extended Data Fig. 1-4), other changes in simple regular graphs have more dramatic effects (Fig1a v Fig1c), displaying different degrees of linear v. non-linear behaviour for different perturbations. In low algorithmic-content networks such as *simply directed complete graphs*, all nodes are immune to perturbations, leaving the basins of attraction and number of attractors the same (only proportional to their new size). From these principles, it is evident that systems that are far from random display inherent regular properties, and are thus more robust in the face of random perturbations because they have deeper attractors (See Supplement Section 2).

Algorithmic Causal Reconstruction of Dynamic Systems

The theory of algorithmic complexity provides means to find mechanistic causes through most likely (simplest) algorithmic models, helping to reverse engineer partial observations from dynamic systems and networks.

The causal reconstruction method of a system (e.g. a network or cellular automaton) M is as follows:

1) Estimate the information contribution of every element e in O(n), the sequence of instantaneous observations O from time 0 to n.
2) The set of neutral elements {e} is the set of those elements whose algorithmic-information content contribution to the complexity O(n) is of a logarithmic nature only with respect to C(n).
3) Remove neutral elements {e} from O(n) and repeat (1) with reassigned O(n) := O(n)\{e}.
4) After m iterations the reverse sequence of observations O(n)\{e} provides an indication of the evolution of the system in time, thereby yielding a hypothesis about the generating mechanism P producing O(n) for any n, and unveiling the initial condition in the last element of the above iteration, or the first after reversing it (see section 2 for more details and an example).

Section 2: Parameter-free and Unsupervised Algorithms: pseudo-codes and evaluations

Dynamical simulations by Boolean networks

We explored whether the algorithmic content, or more precisely the information spectrum, of a system/network, influences transitions between different stable states, thereby effectively

providing a tool with which to steer and reprogram networks. We observed an average decrease in the size of reachable states for all nodes (mean value), and the distribution of reachable states becomes more clustered (standard deviation), and more symmetrical (skewness) for all graphs with five nodes and single deletion. Positive info nodes had a similar effect as the deletion of a hub in the network. Absolute and relative negative nodes have a similar effect, whereas neutral (no information change) nodes preserve the distribution skewness closest to the original.

Histograms of perturbation effects on all graphs of size five nodes using functions XOR, AND, and OR produced similar results (see Fig3d & raw data infoedgesmotifs5.csv). Due to the small size of the graph, we were able to control for graph automorphisms to correct minor BDM errors produced by boundary conditions(Zenil *et al.*, 2016). Two objects x, y are in the same orbit if there is an automorphism φ in Aut(G) such that $\phi(X) = Y$ (equivalently, $X = \phi - 1\ (Y)$). In the algorithmic perturbation analysis, if elements e_1, …, e_n in E are in the same orbit in Aut(G) we take the perturbation of every element in E to be equal to $min\{|\ C(G\backslash e_1) - C(G)\ |, \ldots, |\ C(G\backslash e_n) - C(G)\ |, \ldots\}$. In other words, the effect of every element e_i in E on G is the same. The automorphism group Aut(G) was generated with the help of public software(Brendan D McKay, 1981; Mckay and Piperno, 2014) for this experiment. For larger networks, however, this becomes computationally expensive, in the context of the perturbation analysis, and thus, because we have shown that K(G) ~ K(Aut(G))[26 (main text)], we continued calculating C(G) only.

In the exhaustive experiment over all connected graphs of node count 5, deleting the largest versus smallest node degree produced statistical differences as expected and previously suggested[9]. More relevant to our purposes, it was found that positive versus negative versus neutral information node/edge removal led to statistically different effects when executed in connected networks. Negative information node removal was interestingly not similar to lowest degree removal, yet significantly different statistically from control (random) node removal. Absolute and relative negative information removal had similar effects, and neutral (no information change) nodes/edges kept the distribution skewness closest to the original distribution, in accordance with the theory. For negative edges, the number of attractors was significantly increased (Fig5d), as the theory predicted.

Minimal Information Loss Sparsification (MILS)

Below, we provide the pseudo-code for the MILS algorithm. MILS allows dimensionality reduction of a graph (or any object) by deletion of neutral elements, thus maximizing preservation of the most important properties of an object as the algorithmic information content is invariant under neutral node perturbation. Let G be a graph. Then:

1. Calculate the *powerset spectra*(G) and let E_j be the subset j in the set of all non-empty proper subsets of edges $\{e_1, \ldots, e_n\}$ in G.
2. Remove the subset E_j such that $C(G\backslash E_j) < |C(G\backslash E_i)|$ for all E_i in *powerset spectra*(G) (see Glossary section 1), 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 time complexity class is in $O(2^{p(n)})$ (if there are no subsets with the same information value) because of the combinatorial explosion of the power set, but a more efficient

suboptimal version of MILS iterates only over singletons:

1. Calculate $G\backslash e_i$ for all $i \in \{e_1,...,e_n\}$ i.e. *spectra*(G).
2. Remove edge e_j in *spectra*(G) 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 information content (in particular, it minimizes information loss or introduces spurious information) to the network according to the information difference when removed from the original network.

Assuming that the estimations of C(G) and *spectra*(G) are definite and fixed (in reality one can always find tighter upper bounds, though, due to C's semi-computability), and MILS is a deterministic algorithm. Let G be a network and $i(e) = C(G) - C(G\backslash e)$ be the information value of element e in G with respect to G. If $i(e') > i(e)$ then MILS algorithm removes e first (by definition) because it minimizes the loss of information if the choice is to remove either e or e'. Thus we have it that $C(G\backslash e_1) = C(G\backslash e_2)$ if, and only if, $i(e_1) = i(e_2)$. However, it does not hold in general that $C(G\backslash e_1\backslash e_2) = C(G\backslash e_2\backslash e_1)$, that is, the removal of e_1 followed by the removal of e_2 from G, is not equal to the removal of e_2 followed by the removal of e_1 from G, even for $i(e_1) = i(e_2)$, because of non-linear effects (i.e. the removal of e_i may modify the information contribution of all other e_j in $G\backslash e_i$)**.** This suggests that the only way to deal with these cases for MILS to be deterministic is the simultaneous removal of the set of elements $\{e_1, \ldots e_n\}$ such that $i(e_1) = \ldots = i(e_n)$. The time complexity of MILS thus ranges between the original $O(n^2)$ in the worst case to O(1), when all nodes have the same information value/contribution to G and are thus removed in a single step. Therefore, set removal turns MILS into a proper deterministic algorithm that yields the same object for any run of MILS over an object G. Because any property of a network ultimately contributes to its information content (the amount of information to describe it), information minimization will preserve any potential measure of interest. We show in the following section that minimizing loss of information maximizes the preservation of graph theoretic properties of networks such as edge and node betweenness, clustering coefficient, graph distance, degree distribution and finally information content itself (Fig S5-S6).

Maximal Algorithmic Randomness Preferential Attachment (MARPA) algorithm

MARPA allows constructions of a maximally random graph (or any object) by filling out the blanks, i.e. adding edges, for any given graph in such a manner, that randomness increases. Let G be a network and C(e) the information value of e with respect to G such that $C(G) - C(G\backslash e) = n$. Let $P = \{p_1, p_2, \ldots, p_n\}$ the set of all possible perturbations. P is finite and bounded by $P < 2^{|E(G)|}$ where E(G) is the set of all elements of G, e.g. all edges of a network G. We can find the set of perturbations e' in P such that $C(G) - C(G\backslash e') = n'$ with $n' < n$. As we iterate over all e in G and apply the perturbations that make $n' < n$, for all e, we go through all $2^{|E(G)|}$ possible perturbations (one can start with all $|E(G)|$ single perturbations only) maximizing the complexity of $G' = \max\{G \mid C(G) - C(G\backslash e) = \max$ among all p in P and e in G$\}$. Alternatively, there is a configuration of all edges in G that maximizes the algorithmic randomness of G. Let such a maximal complexity be denoted by maxC(G). Then we find the sequential set of perturbations {P} such that maxC(G) - C(G) = 0, where C(G) - maxC(G) is a measure, related to randomness deficiency,(Buhrman *et al.*, 1999; Antunes *et al.*, 2009) of how removed G is from its (algorithmic-) randomized version maxC(G) (notice that C(G) is upper bounded by maxC(G), and

so the difference is always positive). Fig. 3a-c, shows how we numerically (single-element wise) moved a regular network towards randomness (in particular an E-R graph). Notice that while an ER network with edge density 0.5 is of maximal entropy, it can be of high or low algorithmic randomness, i.e. recursively generated or not,(Zenil and Kiani, 2016) but a high algorithmic-random graph is also ER because, if not, then by contradiction it would be statistically compressible and thus non-algorithmic random, because a graph with any statistical regularity cannot also be an algorithmic-random or an ER graph. One can also consider the absolute maximum algorithmic-random graph, denoted by amaxC(G) and disconnected from the number of elements of G (thus not a randomization of G), that is, the graph comprising the same number of nodes, but an edge arrangement such that C(G) < C(amax(G)) $\leq$ 2^k where $k = (|E(G)|(|E(G)|-1))/4$ is the maximum number of edges in G divided by 2 (at edge density 0.5 it reaches max algorithmic randomness. The process approximates a network of a size that has the greatest possible algorithmic randomness and is also an Erdős-Rényi (ER) graph. The pseudo-code is as follows:

1. Start with graph G (can be empty).
2. Attach edge e_j to edge e_j' in G such that $C(G \cup e_j') > C(G)$.
3. Repeat 1 with $G := G \cup e_j$' until final target size of graph is reached.

Generating a MAR graph is computationally very expensive with time complexity $O(2^{n^2})$ because at every step all possible attachments have to be tested and evaluated (i.e. all possible permutations of the adjacency matrix of size n×n), but small MAR graphs are computationally feasible, and they represent approximations of "perfect" ER random graphs, but unlike some ER graphs they cannot, in principle, be recursively generated with small computer programs. The intuition behind the construction of a MAR graph is that the shortest computer program (measured in bits) that can produce the adjacency matrix of the MAR graph, is of about the size of the adjacency matrix and not significantly shorter. Thus it can in some strict sense be considered the perfect ER graph. Every time that a larger graph, and therefore the addition of new edges, is needed, the computer program that generates it grows proportionally to the size of the adjacency matrix (See Supplement section 1 for algorithm and more details).

Algorithmic Causal Reconstruction of Dynamic Systems

There are systems whose internal kinetics fully determine the system's behaviour, i.e. attractor structure, such as Hopfield networks[13] and Boltzmann machines[14], which is independent of their fixed topology (complete graphs). Other networks are, however, more dependent on topology or geometry (e.g. disease networks or geographical communication networks). Boolean networks are governed both by their topological and internal kinetic properties as encoded by the connectivity of the node with the assigned Boolean function(Kauffman, 1969; Aldana, 2003b) to that very node. Each observation of a system is necessarily only a partial snapshot of the system's trajectory in phase space and it reveals only certain aspects of the generating cause, yet without any loss of generality one can use the causal calculus introduced here either on T, D or a combination of T and D in order to produce algorithmic models of causal generating mechanisms approaching P and producing T and D (see Fig.2M). While the focus of the causal calculus introduced here is on T, it can readily incorporate D by moving to the phase space without any essential modification. We have included some examples using discrete dynamic systems such as cellular automata to show how the same calculus can be utilized. The same

elements in D that move a system towards, or away, from randomness are, conversely, positive and negative elements like those defined for T (see Fig2a-d) in the application to networks.

Reverse-engineering discrete dynamical systems from disordered observations*:*
A cellular automaton (CA) is defined by a rule for computing the new value of each position in a configuration based only on the values of cells in a finite neighborhood surrounding a given position. Commonly a CA evolves on a square grid or lattice of cells updated according to a finite set of local rules which are synchronously applied in parallel. A snapshot in time of the symbols of the cells is called a configuration. A snapshot in space and time (the characteristic CA grid) is called evolution.

A local and a global function f and λ can, therefore, define a cellular automaton. Let S be a finite set of symbols of a cellular automaton (CA). A finite configuration is a configuration with a finite number of symbols, which differs from a distinguished state b (the grid background) denoted by $0^{\infty}b0^{\infty}$ where b is a sequence of symbols in S (if binary then S = {0, 1}). A stack of configurations in which each configuration is obtained from the preceding one by applying the updating rule is called evolution. Formally, Let $f : S^Z \rightarrow S^Z$ where Z is the set of positive integers and n, i ∈ N then $f(rt) = \lambda(xi - r \ldots xi \ldots xi + r)$, where f is a configuration of the CA and r_t a row with t ∈ N and r_0 the initial configuration (or initial condition). The function f is also called the global rule of the CA, with $\lambda : S^n \rightarrow S$ the local rule determining the values of each cell and r the neighborhood range or radius of the cellular automaton, that is, the number of cells taken into consideration to the left and right of a central cell x_i in the rule that determines the value of the next cell x.
All cells update their states synchronously. Cells at the extreme end of a row must be connected to cells at the extreme right of a row for f to be considered well defined. The function λ indicates the local state dependency of the cellular automata and f updates every row. Depicted (FigS4) is the Elementary Cellular Automaton (ECA) rule 254 (in Wolfram's enumeration[17]) that generates a typical 1-dimensional cone from the simplest initial condition (black cell) running downwards over time for 20 steps. ECA is CA that consider only the closest neighbours to the right and left and itself, thus three cells, each with a binary choice for λ. Every ECA such as a rule 254 can thus be seen as a 2^3 = 8-bit computer program represented by its rule icon representing its function f (Fig1a P(t)) or a function determining its local and global dynamics (Fig.1M D(t)). Any perturbation of the simple evolution of the rule leads to an increase of its complexity because a rule with a longer description than rule 254 would be needed to incorporate the random perturbation introduced (blue rows). Thus every row in rule 254 is information negative, except for the random rows whose deletion would bring the rule to its simplest description (rule 254). Unlike the rest of the dynamic system, the last step in the evolution of a dynamic system is information neutral because it does not add or remove any complexity, so removal of neutral elements reverses the system's unfolding evolution to its original cause (the black cell) and the rule can be derived by reversing the sequence of the neutral elements at every step, effectively peeling back the dynamic system from a single instant of a sequence of observations (in optimal conditions, e.g. no noise and full accuracy, and good enough approximations of algorithmic-information content).
When clustering consecutive rows of the evolution of all Elementary Cellular Automata (256 rules), we found that the later the perturbations in time, the more neutral, thus conforming to the theoretical expectation (Fig. 3 main texts). When taking a sample of representative ECA, this was also clearly the case (Fig. 3 main texts). We proceeded to reverse engineer the rule of an ECA by:

1) Producing the space-time diagram O(n) of an ECA from time 0 (initial condition) to time n.
2) Scrambling the observations from O(n) (worse case of an observation, to lose track of their order)
3) Sorting the scrambled observations to maximize the algorithmic probability and thus find the most likely generating mechanism (with lowest algorithmic complexity).
4) From 2 and 3 estimating the algorithmic-information content of every (hypothesized) step.
5) Comparing among them and sorting from the lowest contribution to the highest.
6) Finding the initial condition and generating rule by reversing the order of the sequence of neutral elements from O(n).

Finding the lowest complexity configuration of disordered observations, we show how we found the correct times, thus generating a most powerful method to reverse engineer and find design principles and the generating mechanism of evolving systems. Running the sequence forward, one can also make predictions about the phase space configuration of the dynamic evolving system. Fig. 2 shows that the predicted point in the phase space does not diverge from the actual position of the system in phase space, thus providing good estimations of the evolution of the system both backward and forward.

In this paper, we choose to work at the level of T (see Fig. 2 main texts) for the same convenient simplifying reasons followed by other network-based approaches, but unlike other possible approaches, the theory and methods hold in general for non-linear dynamical systems and not only for static or evolving networks. When working on T only, we assume that lossless descriptions are of the observations (e.g. only T) and not of full descriptions of T and D or even P (the true generating mechanism, e.g. a computer program P) that is the unknown. To date, there have been no other alternatives to applying non-linear interventions to complex systems in the phase space other than to calculate the dynamical properties of a system, often assumed with little knowledge or else assumed to be linear and in fixed states, requiring computationally intractable simulations. This new calculus, however, requires much less information to make educated causal interventions that prove to be extremely useful and powerful.

Entropy-deceiving graphs

We introduced a method[6 (main text)] for building a family of recursive graphs of which one is denoted by 'ZK' with the property of being recursively constructed and thus of low algorithmic (Kolmogorov-Chaitin-Solomonoff) complexity (hence causal) but that to an uninformed observer would appear statistically random and thus as having maximal Entropy. These graphs were proven to have maximal Entropy for some lossless descriptions but minimal Entropy for other lossless descriptions of exactly the same object, thereby demonstrating how Entropy fails at unequivocally and unambiguously characterizing a graph independent of a particular feature of interest reflected in the choice of natural probability distributions. A *natural probability distribution* of an object is given by the uniform distribution suggested by the object dimension and its alphabet size. For example, if a graph G is losslessly (with no loss of information) described by its adjacency matrix M, then in the face of no other information, the natural distribution is the probability space of all matrices of dimensions |M| and binary alphabet. If, however, G is losslessly described by its degree sequence S, with no other information provided about G, the natural distribution is given by the probability space of all sequences of length |S| and alphabet size |{S}|, where {S} denotes the number of different n-ary symbols in S. The

natural distribution is thus the less informative state of an observer with no knowledge of the source or nature of the object (e.g. its recursive character). We denote by 'ZK' the graph (unequivocally) constructed as follows:

1. Let 1 → 2 be a starting graph G connecting a node with label 1 to a node with label 2. If a node with label n has degree n, we call it a *core node*, otherwise, we call it a supportive node.
2. Iteratively add a node n + 1 to G such that the number of core nodes in G is maximized.
3. The resulting graph is typified by the one in Fig3c in the main text.

Supporting nodes are always the latest to be added at each iteration. Perturbing elements of the network other than the last elements will break the generating program and thus these elements will move the network towards randomness, whereas removing the latest nodes has little to no impact because it only moves the network back in time, the originating program remaining the same and only needing to run again to reach the same state as before. Thus by inspecting elements that do not contribute or make the network slightly simpler, one can reverse the network in time, thereby revealing its generating mechanism (See the subsection titled Algorithmic Causal Reconstruction of Dynamic Systems).
We have shown that Entropy is highly observer dependent[6 (main text)], even in the face of full accuracy and access to lossless object descriptions. For these specific complexity-deceiving graphs Entropy produces disparate values when the same object is described in different ways (thus with different underlying probability distributions), even when the descriptions reconstruct exactly the same, and only the same, object. This drawback of Shannon Entropy, ultimately related to its dependence on distribution, is all the more serious because it is often overlooked for objects other than strings, such as graphs. For an object such as a graph, we have shown that changing the descriptions may not only change the values but that divergent, contradictory values are produced. This means that one not only needs to choose a description of interest to apply a definition of Entropy-- such as the adjacency matrix of a network (or its incidence or Laplacian) or its degree sequence--but that as soon as the choice is made, Entropy becomes a trivial counting function of the feature-- and only the feature--of interest. In the case of, for example, the adjacency matrix of a network (or any related matrix associated with the graph, such as the incidence of Laplacian matrices), Entropy becomes a function of edge density, while for degree sequence, Entropy becomes a function of sequence normality. Entropy can thus trivially be replaced by such functions without any loss, but it cannot be used to profile the object (randomness, or information content) in any way, independent of an arbitrary feature of interest. The measures introduced here are robust measures of (graph) complexity independent of object description based upon the mathematical theory of randomness and algorithmic probability (that includes statistical randomness), which are sensitive enough to deal with causality and provide the framework for a causal interventional calculus.

Section 3: Evaluation and validation of the causal calculus using transcriptional data and genetic regulatory networks.

E-Coli Transcription Factor Network Ontology Enrichment Analysis
We estimated the information node values of a highly curated E. coli transcriptional network (only experimentally validated connections) from the RegulonDB (http://www.ccg.unam.mx/en/projects/collado/regulondb). Info values were clustered into 6

clusters by using partitioning around K-medoids and optimum average silhouette width. Gene clusters were tested for enrichment of biological functions according to Gene Ontology, KEGG and EcoCyc databases, using the topGO "weight01" algorithm for GO or hypergeometric enrichment test for KEGG and EcoCyc. BDM values did not correlate with degree distribution, compression or Shannon entropy. The numerical results suggest that more positive information genes in E-Coli are related to homeostatic processes, while more negative info genes are related to processes of specialization, which is in agreement with the idea that cellular development is an unfolding process in which core functions are algorithmically developed first, then more specialized functions, enabling training-free and parameter-free gene profiling and targeting. Figures S8-S16 show that other measures fell short at producing statistically significant groups for a gene ontology analysis, and also provide details of the clusters found and the elements comprising them.

Information spectral and enrichment analysis of Th17 differentiation

We applied our method to a dataset on differentiation of T-helper 17 (Th17) cells. Th17 cells are one of the major subsets of T-helper cells, which in addition to Th17 comprise several sub-types such as Th1, Th2 and Treg cells. These subsets all differentiate from a common naïve CD4+ T cell precursor cell type based on environmental signals and are classified by certain lineage-defining markers. Th17 cells are necessary to protect the host from fungal infections, but at the same time are involved in the pathogenesis of several autoimmune diseases, hence the processes driving Th17 differentiation are of great interest to the scientific community[20]. From the gene ontology analysis taking the experimentally known genes involved in the process of differentiation from T naïve to Th17 (Fig. 3 e), it is shown that precisely these genes are distributed non-uniformly and in different ways along the 3 time points, suggesting that the algorithmic perturbation analysis succeeds at identifying such genes (otherwise, the distributions would have appeared uniform in all cases).

Information spectral analysis

The information spectral analysis used a reconstructed regulatory network from functional perturbation and transcriptional data corresponding to the Th17 differentiation. The data was divided into 3 time windows: 0.5 to 2 hours, 4 to 16 hours, and 20 to 72 hours, here referred to as EarlyNet, IntermediateNet and FinalNet respectively. We were interested in investigating whether genes with strongly negative or positive information values would include genes known to be crucial in Thelper cell differentiation and/or novel putative Th17 regulators, and whether these genes would, according to our predictions, change their information content throughout the Th17 differentiation process. We noted that in general, genes classified as having the most positive or negative information values covered several genes known to be involved in T cell differentiation, such as transcription factors from the IRF or STAT families (see Figure S5). The genes assigned to the Th17 regulating modules[19] were present along the spread of information values, with some enrichment at extreme positive values. However, not all genes with extreme information values were identified in the original study,[35(main text)] suggesting that our method may identify additional regulators Figure S5). When analyzing those genes that are present in all 3 networks and determining their evolution over time (Figure S5), we noted that genes for chemokines/chemokine receptors were switching from negative values in EarlyNet to positive values in FinalNet. In the gene group switching from positive in EarlyNet via negative in IntermediateNet back to positive in FinalNet, many transcription factors from the STAT family were represented. Extreme (mostly positive) information values were assigned to many members of the IRF family of transcription factors, which comprises well-known regulators of

Thelper differentiation (Figure S5), including Th17-inhibiting roles for IRF8 which appears at the top of the lists in IntermediateNet and FinalNet (Figure S5). Only three genes were assigned negative information values in FinalNet, namely STAT6, TCFEB and TRIM24, suggesting that removing these might enhance the Th17 profile.

Clustering

The networks were clustered using the k-means algorithm with 5 clusters per network (Figure S5). The list of genes that changed from most negative information values in EarlyNet (cluster 5) toward most positive information values in FinalNet (cluster 1) contained several genes involved in T helper cell subset differentiation and function, for example, HIF1a, FOXO1, IKZF4, IL2, IL21, IL2RA, IL6ST. Conversely, the list of genes with the highest information values in EarlyNet overlapping with the lowest information values in FinalNet was more restricted in number and contained some general transcription factors such as RelA and Jun.

We noted that in general, genes classified as having the most negative or positive information values comprised many genes known to be involved in T cell differentiation, such as transcription factors from the IRF or STAT families, chemokine receptors, cytokines and cytokine receptors. This was particularly evident for networks 1 and 3. When analyzing those genes that have negative information values in network 1 and that change towards positive information values in network 3, we found that the common elements in both lists contain several such genes involved in T helper cell subset differentiation and function, for example, HIF1a, FOXO1, IKZF4, IFNg, IL2, IL21, IL2RA, IL6ST, CXCL10, CXCR3, CXCR5. Interestingly, the list of genes with positive information values in network 1 or with negative information values in network 3 was much more restricted in number and did not overlap, yet contained highly interesting genes. In network 1, these were mostly transcription factors, including several IRFs, STATs as well as RUNX1 and SMAD2, all known to be important in T cell differentiation. The few genes with negative information values in network 3 were STAT6, TCFEB and TRIM24 (interestingly these 3 genes, STAT6, TCFEB, TRIM24 are amongst the few centered around 0, i.e. neutral, in network 1), and it is tempting to speculate that over-activation of these might be able to reprogram differentiated Th17 cells to another lineage. Indeed, STAT6 is a well-known factor in IL-4 response and Th2 induction. Notably, in network 2, which may be viewed as a transition state, 3 genes were assigned the most positive information values and all of these belonged to the IRF family of transcription factors, which comprises well-known regulators of Thelper differentiation, including Th17-inhibiting roles for IRF8 which appears in said list.

Enrichment Analysis

To assess to what extent our informational spectral analysis identifies genes, which are relevant to the differentiation process in Th17 cells, we perform an enrichment analysis based on a literature survey. To this end we collected 9 landmark papers in the Th17 literature (Carneiro, Chaouiya and Thieffry, 2010; Ghoreschi *et al.*, 2010; Zhu, Yamane and Paul, 2010; Hong *et al.*, 2011; Ciofani *et al.*, 2012; Lee *et al.*, 2012; Tuomela *et al.*, 2012; Yosef *et al.*, 2013; Gaublomme *et al.*, 2015)[28].

From each paper a list of genes was extracted (manually), in an attempt to select the set of genes, which the text identified as relevant to Th17 differentiation. The script calculates all the intersections between these sets, with genes at a greater number of intersections given a higher weight as being more relevant in the Th17 literature (the list of genes is in Sup. file output_with_kuchroo.txt). The data is represented in a network diagram (Figure S8-S13) where a co-occurrence analysis highlighted genes that were commonly identified across several studies.

The enrichment analysis revealed that positive and negative information elements were not distributed equally, thus indicating that information values were not distributed by chance in any of the three time steps, and that these changed over time according to the theoretical and biological expectations. That is, at early stages the naïve cell has two strong sets of genes that act as handles to steer the network towards or away from randomness, with a larger component of negative elements that indicate signals that are either activating the cells or perturbing cells among the stable naïve cells that are key to the original (undifferentiated steady state) program. Then cells are activated and fewer negative genes are present, while there is a distribution skew of the positive patch towards neutral elements that pinpoint the evolving genes from the cell activation for differentiation (high peak in the enrichment analysis). At the final step, the cells no longer have negative elements, indicating that the program has reached a steady state and the cells have been fully differentiated, with all remaining elements either positive or closer to neutral.

CellNet Waddington landscape

CellNet is a network biology-based computational platform that assesses the fidelity of cellular engineering and claims to generate hypotheses for improving cell derivations(Cahan *et al.*, 2014)[(main text)]. We merged networks of the same tissue type into a single larger entity. The result led to a set of networks of networks of the following 16 Homo Sapiens cell types: B-cell, colon, endothelial, esc (embryonic stem cell), fibroblast, heart, hspc (Hematopoietic stem cells), kidney, liver, lung, macrophage, muscleSkel, neuron, ovary, skin and tcell, each with the following vertex count: 12006, 4779, 5098, 16581, 8124, 6584, 21758, 5189, 4743, 1694, 5667, 6616, 10665, 1623, 3687 and 11914, on which we applied the causal calculus and reprogrammability measures (SI Section 1).

REFERENCES

Aldana, M. (2003a) 'Boolean dynamics of networks with scale-free topology', 185(May), pp. 45–66. doi: 10.1016/S0167-2789(03)00174-X.

Aldana, M. (2003b) 'Boolean dynamics of networks with scale-free topology', *Physica D: Nonlinear Phenomena*, 185(May), pp. 45–66. doi: 10.1016/S0167-2789(03)00174-X.

Antunes, L. *et al.* (2009) 'Depth as Randomness Deficiency', pp. 724–739. doi: 10.1007/s00224-009-9171-0.

Brendan D McKay (1981) 'Practical graph isomorphism', *Congressus Numerantium*, 30, pp. 45–87.

Buhrman, H. *et al.* (1999) 'Kolmogrov Random Graphs and the Incompressibility Method', *Society for Industrial and Applied Mathematics*, 29(2), pp. 590–599.

Cahan, P. *et al.* (2014) 'CellNet: Network Biology Applied to Stem Cell Engineering', *Cell*, 158(4), pp. 903–915. doi: 10.1016/j.cell.2014.07.020.

Carneiro, J., Chaouiya, C. and Thieffry, D. (2010) 'Diversity and Plasticity of Th Cell Types Predicted from Regulatory Network Modelling', 6(9), pp. 9–12. doi: 10.1371/journal.pcbi.1000912.

Chaitin, G. (1987) *Algorithmic Information Theory*. Cambridge University Press.

Ciofani, M. *et al.* (2012) 'A validated regulatory network for Th17 cell specification', *Cell*. 2012/10/02, 151(2), pp. 289–303. doi: 10.1016/j.cell.2012.09.016.

Espanés, P. M. De, Osses, A. and Rapaport, I. (2016) 'BioSystems Fixed-points in random Boolean networks : The impact of parallelism in the Barabási – Albert scale-free topology case ☹',

BioSystems. Elsevier Ireland Ltd, 150, pp. 167–176. doi: 10.1016/j.biosystems.2016.10.003.
Gaublomme, J. T. *et al.* (2015) 'Single-Cell Genomics Unveils Critical Regulators of Th17 Cell Pathogenicity', *Cell*. Elsevier Inc., pp. 1–13. doi: 10.1016/j.cell.2015.11.009.
Ghoreschi, K. *et al.* (2010) 'Generation of pathogenic T(H)17 cells in the absence of TGF-β signalling.', *Nature*, 467(7318), pp. 967–71. doi: 10.1038/nature09447.
Hong, T. *et al.* (2011) 'A mathematical model for the reciprocal differentiation of T helper 17 cells and induced regulatory T cells.', *PLoS computational biology*, 7(7), p. e1002122. doi: 10.1371/journal.pcbi.1002122.
Kauffman, S. a (1969) 'Metabolic stability and epigenesis in randomly constructed genetic nets.', *Journal of theoretical biology*, 22(3), pp. 437–467. doi: 10.1016/0022-5193(69)90015-0.
Lee, Y. *et al.* (2012) 'Induction and molecular signature of pathogenic TH17 cells.', *Nature immunology*, 13(10), pp. 991–9. doi: 10.1038/ni.2416.
Mckay, B. D. and Piperno, A. (2014) 'Practical graph isomorphism, II', *journal of symbolic computation*, 60, pp. 94–112.
Tuomela, S. *et al.* (2012) 'Identification of early gene expression changes during human Th17 cell differentiation.', *Blood*, 119(23), pp. e151-60. doi: 10.1182/blood-2012-01-407528.
Wuensche, A. (2004) 'Basins of attraction in network dynamics : A conceptual framework for biomolecular networks', in Wagner, G. and Schlosser, G. (eds). Chicago University Press, pp. 1–17.
Yosef, N. *et al.* (2013) 'Dynamic regulatory network controlling TH17 cell differentiation', *Nature*, 496(7446), pp. 461–468. doi: 10.1038/nature11981.
Zenil, H. *et al.* (2016) 'A Decomposition Method for Global Evaluation of Shannon Entropy and Local Estimations of Algorithmic Complexity', pp. 1–48.
Zenil, H. and Kiani, N. (2016) 'Low Algorithmic Complexity Entropy-deceiving Graphs'.
Zhu, J., Yamane, H. and Paul, W. E. (2010) 'Differentiation of Effector CD4 T Cell Populations [*]', *Annual Review of Immunology*, 28(1), pp. 445–489. doi: 10.1146/annurev-immunol-030409-101212.

Supplementary Figures

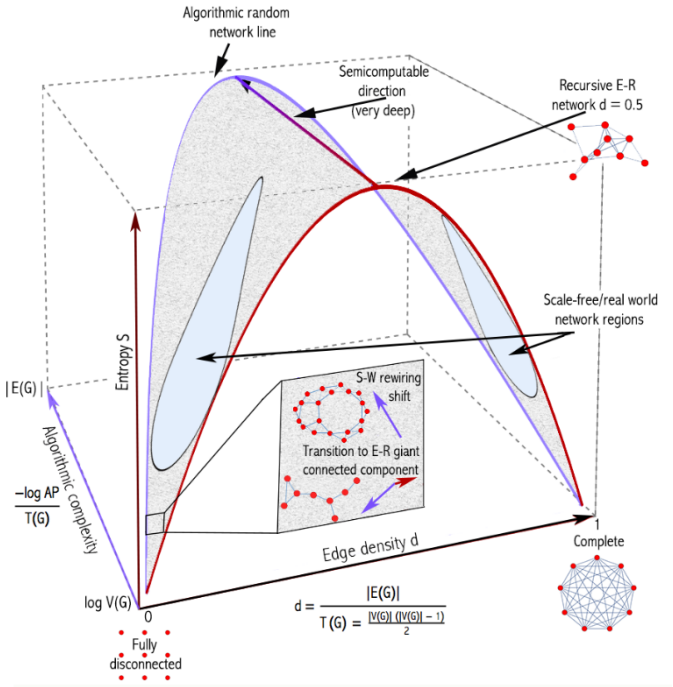

Figure S1, related to Figures 3,4, 5: Algorithmic complexity (numerically approached by way of Algorithmic Probability) adds an additional dimension (depth), complementary but different from the notion of entropy, when performing network analysis. Unlike statistical mechanical approaches such as Shannon Entropy (for strings or networks), algorithmic complexity improves over Entropy by assigning lower Entropy and thus higher causal content to objects that not only appear statistically simple but also algorithmically simple by virtue of having a short generating mechanism capable of reproducing the causal content of a network. Without such an additional dimension, causal and non-causal networks are collapsed into the same typical Bernoulli distribution. Indeed, a random-looking network with maximal Shannon entropy can be recursively generated by a short algorithm that Entropy would misclassify as random. This additional dimension that we introduce in the study of dynamic systems, in particular networks, together with methods and tools, is thus key to better tackling the problem of revealing first principles and discovering causal mechanisms in dynamic evolving systems. The new dimension can account for all types of structures and properties and is sensitive in both directions, where computable or statistical measures would not be. Indeed, an Erdös-Rényi graph, for example, can be recursive or not, with recursivity meaning that it is actually pseudo-random and only has the properties of a random graph but is not algorithmic-random. This distinction is key in science, where evolving systems may be random-looking but are governed by rules that are otherwise concealed by apparent noise.

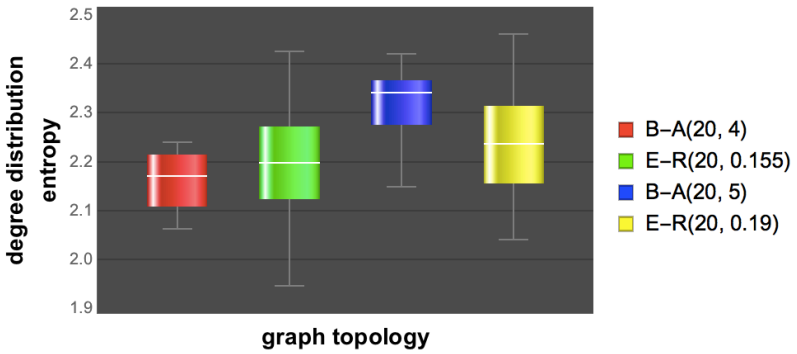

Figure S2, rela: Entropy can easily be fooled. Here is a preferential attachment algorithm (B-A) creating networks of growing density (edge number per node) showing Entropy when calculated on adjacency matrices by only capturing graph density, assigning dense B-A graphs higher entropy than Erdös-Rényi (E-R) graphs. This result was reproduced in 30 replicates using 20 node graphs and 20 replicates/graphs and the experiment was repeated approximately 10 times[29]. The main Fig2c shows another graph created recursively (and thus of low algorithmic complexity) that suggests divergent values of Entropy for the same object but with different descriptions, suggesting different probability distributions. A different, more robust approach to characterizing networks and systems is thus needed to be able to tell these cases apart, moving into the algorithmic mechanics/calculus introduced here and thus
improving over traditional techniques that draw heavily upon statistical mechanics.

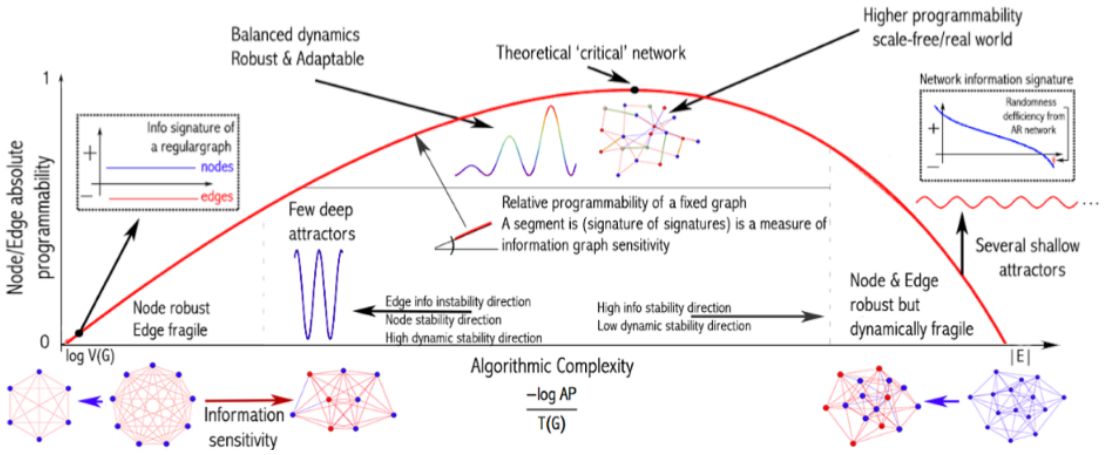

Figure S3 related to Figures 3,4 and 5. A thermodynamic-like effect based on (re)programmability, a measure of sophistication: Moving random networks by edge removal is significantly more difficult than moving simple networks towards randomness. For random graphs, there are only a few elements, if any, that can be used to move it slowly towards simplicity. In contrast, a greater number of elements can move a simple network faster towards randomness. This relationship, described by the reprogrammability rate $\Delta(G) < \Delta(G')$ (see Sup Mat) for G simple and G' random graphs of the same size (vertex count), induces a thermodynamic-like asymmetry based on algorithmic probability and reprogrammability. A MAR graph, which is of the highest algorithmic randomness, has $\Delta(MAR) = \log n$ for all its elements

after n element removals, and thus cannot be easily moved towards greater randomness. This reprogrammability landscape is thus also expected to be related to the dynamical space (epigenetic) landscape with controlled effects in the phase space according to the complexity and the reprogrammability indices of a system, simple connected graphs having fewer attractors than random graphs of the same size. As we have found and reported in the main text and S.I., moving connected networks towards randomness tends to increase the number of attractors (and therefore make them shallower), providing key insights into the epigenetic Waddington landscape and a tool to move systems and networks hitherto impossible to induce to perform in optimal ways other than by actual simulation. Conversely, moving connected networks away from randomness will tend to reduce the number of attractors (and thereby increase the depth of the remaining ones).

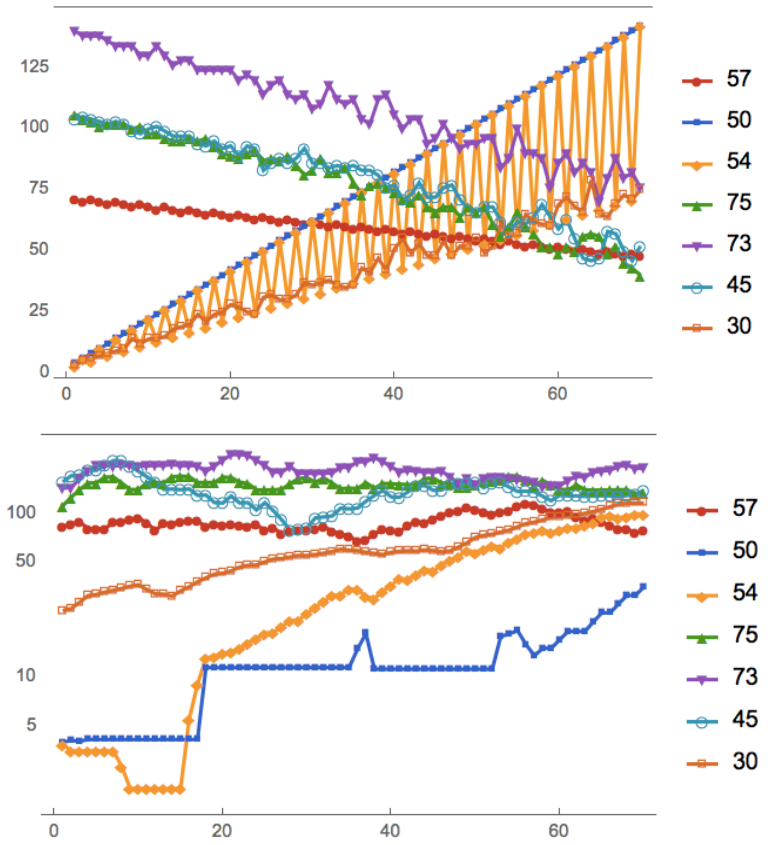

Figure S4 related to Figure 2 and 3 . Qualitative reconstruction by representing each row in a CA as a binary vector, which produces a 2n+1 dimensional phase space, where n is the CA runtime for a sample of representative ECAs. The hamming distance between the binary vectors is used to calculate the behaviour of the moving particle indicating the state of the ECA (top plot). Applying the same procedure to the hypothesized generating mechanism, as identified from our causal calculus, we find that the moving average (bottom plot) of the predicted particle

qualitatively moves in a similar fashion (e.g. increasing v. decreasing/constant) as the original ECA, and the order among the lines corresponds to the original one.

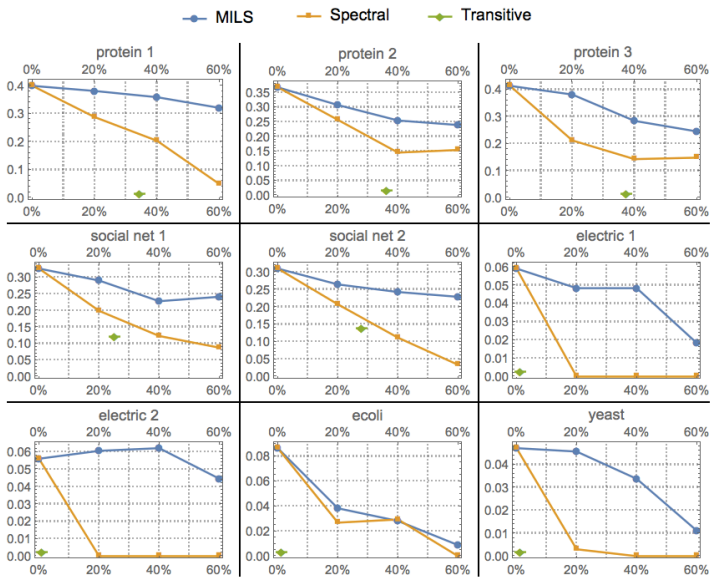

Figure S5 related to Figure 1M **:** Evaluating MILS using nine benchmark networks common in the literature[39] as regards its ability, compared to two state-of-the-art network dimensionality reduction methods, to preserve the clustering coefficient of the original networks while removing up to 60% of all the network edges [40]. Similarly, MILS preserved edge betweenness, degree distribution (see Main Figure 1 l-p) and information signatures (by design) better than mainstream state-of-the-art methods such as transitive and spectral sparsification and null-methods such as random edge/node deletion and lowest degree node deletion (Figure S6). This is to be expected because all these properties of a network are part of its description. MILS thus minimizes the loss of information by maximizing the preservation of all the properties of the original networks.

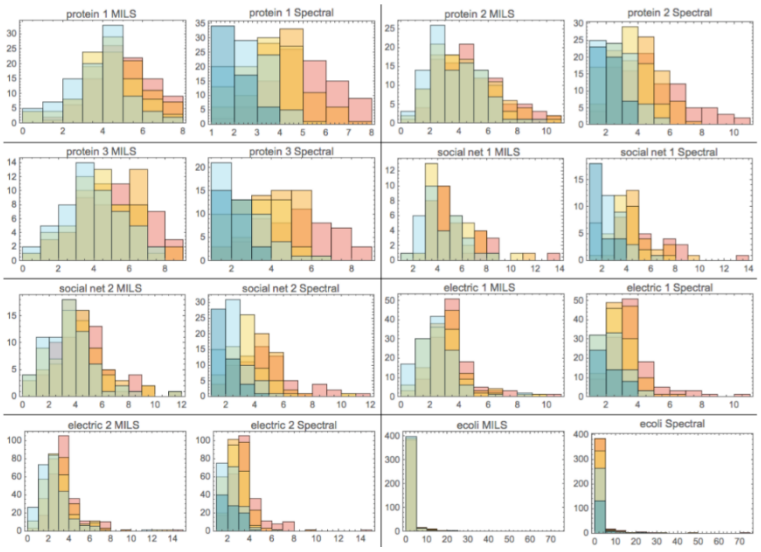

Figure S6 related to Figure 1M**.** Histograms showing the preservation of degree distributions by MILS against a benchmark dimensionality-reduction algorithm based on graph spectra that maximizes the preservation of the graph eigenvalues when removing 20% of the edges (blue), 40% (yellow), 60% (orange) and 80% (pink). The colour green represents the overlapping of areas for each graph and each method. The graphs used are a set of benchmarking graphs in the literature[39].

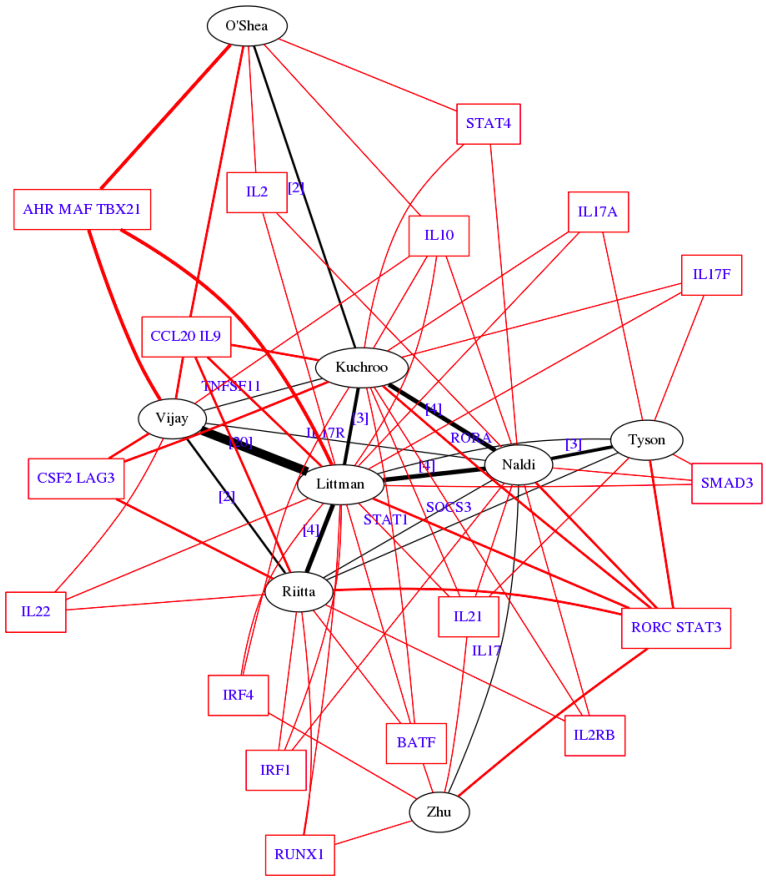

Figure S7 related to Figure 5**:** Network Venn diagram of genes (square nodes) occurring in the 9 major papers in the literature (black elliptic nodes) covering investigations of Th17 cells[30-39]. These papers cover the majority of genes which have been associated with Th17 cells. Linked genes in the figure are genes found in common between two or more papers. Black lines show the number of genes found in common between two papers (with the thickness denoting the size of the overlap). These genes were used in main Figure 4f,g,h in the gene enrichment analysis of the Th17 differentiation network.

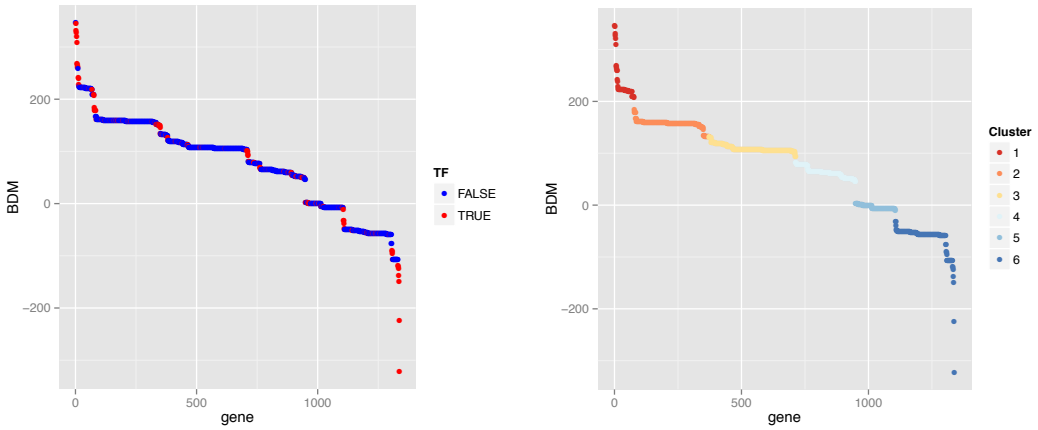

Figure S8 related to Figure 5**.** Six clusters were selected using partitioning around medoid clustering. The number of clusters was estimated by optimum average silhouette width.

	GO.ID	Term	Pval
Cluster 1	GO:0006094	gluconeogenesis	1.60E-06
	GO:0006096	glycolysis	0.00036
	GO:0008615	pyridoxine biosynthetic process	0.0124
	GO:0009255	Entner-Doudoroff pathway	0.0124
	GO:0042330	taxis	0.02035
	GO:0016052	carbohydrate catabolic process	0.02911
2	-	-	-
3	-	-	-
4	-	-	-
5	-	-	-
III	GO:0006793	phosphorus metabolic process	2.10E-08
	GO:0009252	peptidoglycan biosynthetic process	2.90E-07
	GO:0006777	Mo-molybdopterin cofactor biosynthetic process	1.20E-05
	GO:0009086	methionine biosynthetic process	0.0027
	GO:0009242	colanic acid biosynthetic process	0.0124
	GO:0006164	purine nucleotide biosynthetic process	0.0196
	GO:0009228	thiamine biosynthetic process	0.0254
	GO:0009243	O antigen biosynthetic process	0.0254

Figure S9 related to Figure 5**.** Gene Ontology **GO** database (Biological Process category): over-represented categories tested with TopGO weight01 method (Fisher p<0.05)

	KEGG ID	Term	Pval
Cluster 1	00010	Glycolysis / Gluconeogenesis	1.76E-08
	00051	Fructose and mannose metabolism	7.13E-06
	02030	Bacterial chemotaxis	6.32E-05
	02020	Two-component system	7.55E-04
	00620	Pyruvate metabolism	4.08E-03
	00030	Pentose phosphate pathway	5.14E-03
	02060	Phosphotransferase system (PTS)	5.45E-03
	00680	Methane metabolism	6.70E-03
	01110	Biosynthesis of secondary metabolites	9.59E-03
	01120	Microbial metabolism in diverse environments	1.44E-02
2	-	-	-
3			
4	-	-	-
5	-	-	-
6	00550	Peptidoglycan biosynthesis	1.01E-07
	01100	Metabolic pathways	6.74E-04
	04122	Sulfur relay system	4.11E-03
	00621	Dioxin degradation	9.20E-03
	00622	Xylene degradation	9.20E-03
	00360	Phenylalanine metabolism	1.48E-02
	00300	Lysine biosynthesis	2.48E-02
	00230	Purine metabolism	3.50E-02
	00670	One carbon pool by folate	3.73E-02

Figure S10 related to Figure 5**.** Over-represented **KEGG** pathways database (p<0.05)

	EcoCyc pathway	Term
Cluster 1	superpathway of glycolysis and Entner-Doudoroff	5.37E-07
	Sugar Alcohols Degradation	4.82E-06
	superpathway of hexitol degradation (bacteria)	1.91E-05
	glycolysis I (from glucose-6P)	1.91E-05
	glycolysis II (from fructose-6P)	1.91E-05
	gluconeogenesis I	2.56E-04
	Gluconeogenesis	2.56E-04
	Sugar Derivatives Degradation	0.003115401
	Secondary Metabolites Degradation	0.003131693

	superpathway of glycolysis, pyruvate dehydrogenase, TCA, and glyoxylate bypass	0.004830985
	TCA cycle	0.004830985
	Glycolysis	0.005196795
	Generation of Precursor Metabolites and Energy	0.005701038
	sedoheptulose bisphosphate bypass	0.037381258
	Entner-Duodoroff Pathways	0.037381258
	Entner-Doudoroff pathway I	0.037381258
	CpxAR Two-Component Signal Transduction System	0.037381258
	Signal transduction pathways	0.045972995
2	-	-
3	-	-
4	-	-
5	-	-
Cluster 6	methylphosphonate degradation I	9.40E-06
	Phosphorus Compounds Metabolism	9.40E-06
	Methylphosphonate Degradation	9.40E-06
	Pyrimidine Nucleobases Degradation	0.003167986
	Uracil Degradation	0.003167986
	uracil degradation III	0.003167986
	peptidoglycan biosynthesis (meso-diaminopimelate containing)	0.003167986
	Peptidoglycan Biosynthesis	0.003167986
	Cell Wall Biosynthesis	0.003167986
	putrescine degradation II	0.005063846
	3-phenylpropionate and 3-(3-hydroxyphenyl)propionate degradation	0.018877832
	proline to cytochrome bo oxidase electron transfer	0.019695489
	UDP-N-acetylmuramoyl-pentapeptide biosynthesis I (meso-DAP-containing)	0.028546946
	UDP-N-Acetylmuramoyl-Pentapeptide Biosynthesis	0.028546946
	2-oxopentenoate degradation	0.04015748
	Putrescine Degradation	0.0413727
	Pyrimidine Nucleotides Degradation	0.06959294
	superpathway of ornithine degradation	0.075477235
	Purine Nucleotides De Novo Biosynthesis	0.075477235
	superpathway of purine nucleotides de novo biosynthesis II	0.075477235
	superpathway of arginine, putrescine, and 4-aminobutyrate degradation	0.09681385
	L-rhamnose degradation I	0.09815362
	L-rhamnose Degradation	0.09815362

Figure S11 related to Figure 5**.** Over-represented **EcoCyc pathways** (FDR<0.05)

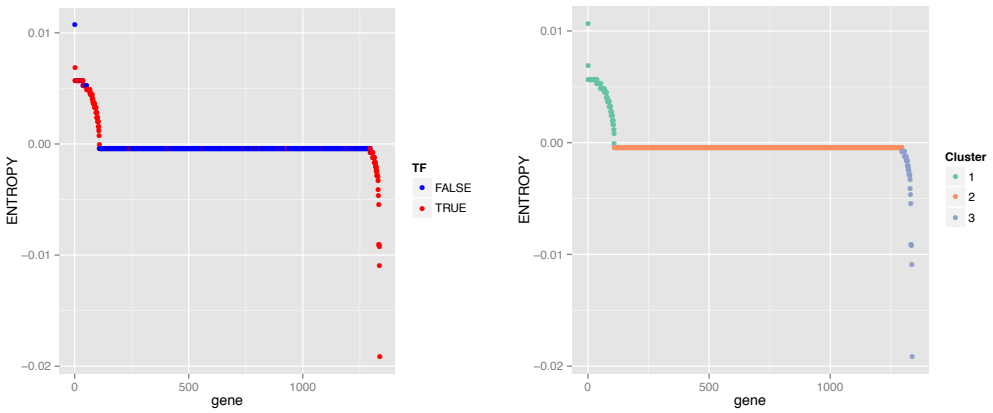

Figure S12 related to Figure 5**.** Three clusters (above baseline, baseline, below baseline) were identified for Entropy which proved to be less sensitive, clustering most elements over the X axis. Non-baseline nodes are enriched for Transcription Factors.

	GO.ID	Term	Pval
Cluster 1	GO:0006805	xenobiotic metabolic process	0.0033
	GO:0009268	response to pH	0.0147
	GO:0006355	**regulation of transcription**, DNA-dependent	0.0298
Cluster 2	GO:0006457	protein folding	0.025
Cluster 3	GO:0009255	Entner-Doudoroff pathway	0.0023
	GO:0009435	NAD biosynthetic process	0.0108

Figure S13 related to Figure 5**.** Gene Ontology (Biological Process): over-represented categories tested with TopGO weight01 method (Fisher p<0.05) using Shannon Entropy. No significant groups were found after GO enrichment analysis.

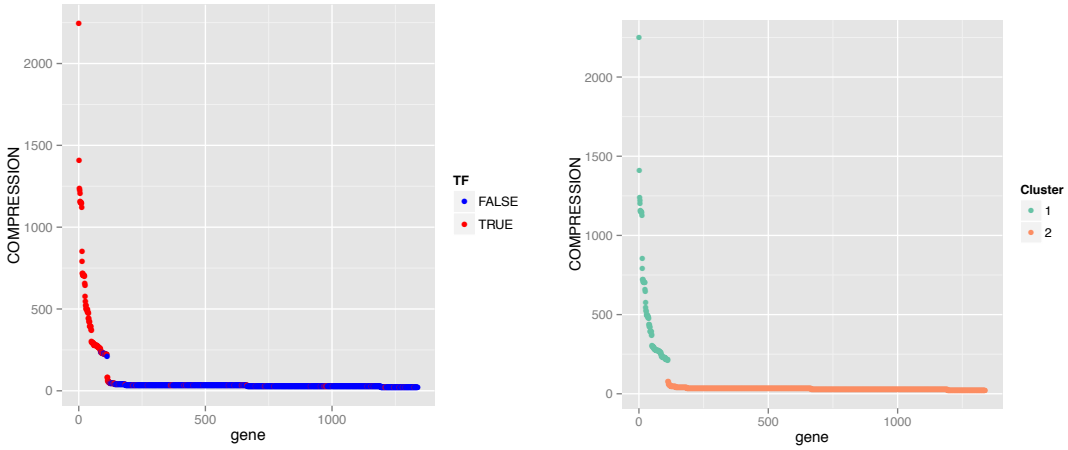

Figure S14 related to Figure 5. Two clusters identified using Compress (above baseline, baseline). Above-baseline nodes are enriched for Transcription Factors. No significant groups were found after GO enrichment analysis.

	GO.ID	Term	Pval
Cluster 1	GO:0006805	xenobiotic metabolic process	0.003
	GO:0009255	Entner-Doudoroff pathway	0.014
	GO:0006355	**regulation of transcription**, DNA-dependent	0.029
Cluster 2	-	-	

Figure S15. Gene Ontology (**Biological Process**): Over-represented categories tested with TopGO weight01 method (Fisher p<0.05) using lossless compression (Compress algorithm).

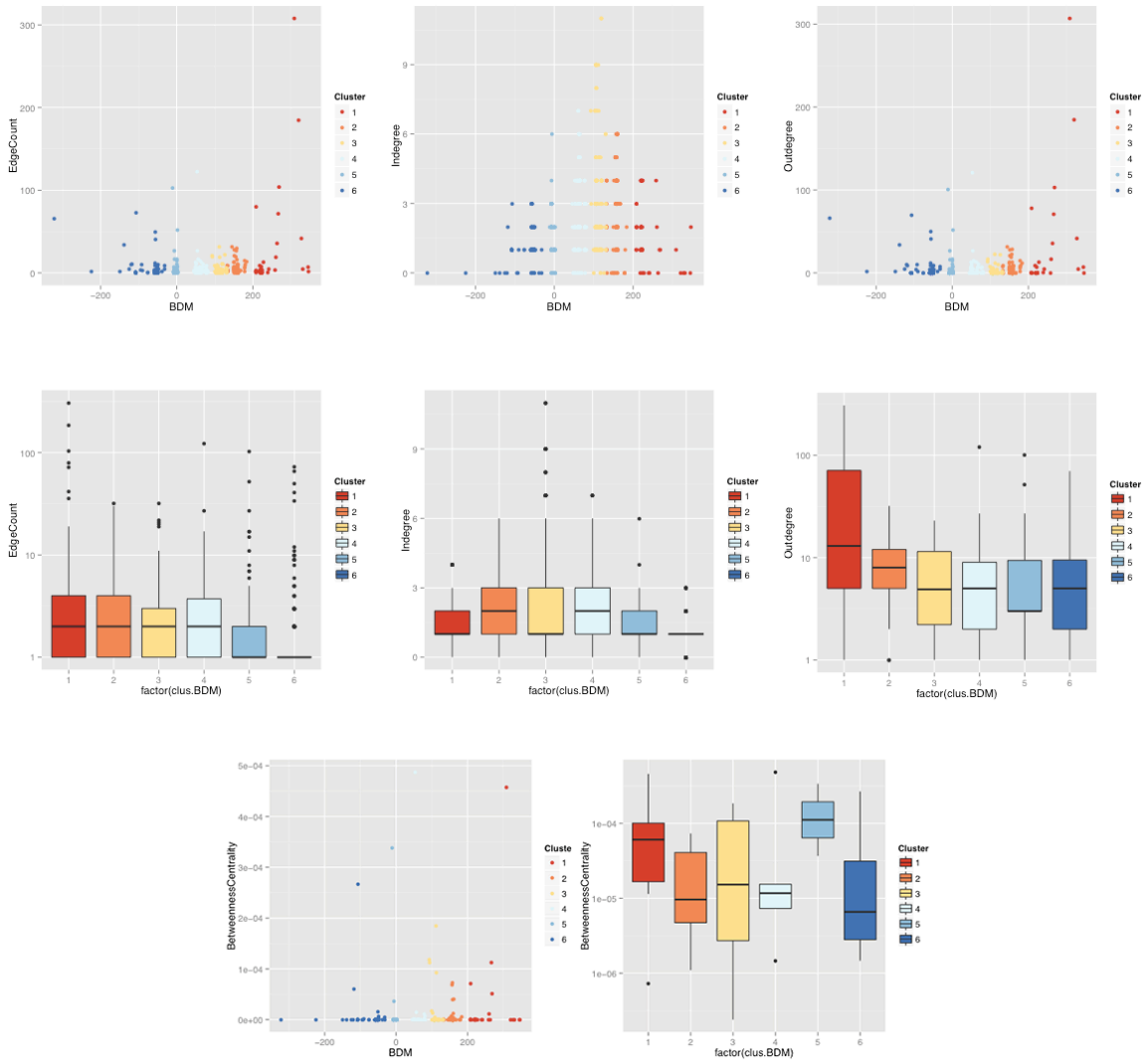

Figure S16 related to Figure 5**.** Unlike graph-theoretic measures that can be described as single or composed functions of other graph-theoretic measures, BDM was not found to correlate with any of these measures, just as it did not correlate with lossless compression and Shannon entropy. Control Experiments: All attempts to produce statistically significant clusters from graph-theoretic measures, lossless compression and Shannon entropy failed when tested against the same Gene Ontology databases.