Exact analytical solution of irreversible binary dynamics on networks

Edward Laurence,[1,*] Jean-Gabriel Young,[1] Sergey Melnik,[2] and Louis J. Dubé[1,†]
[1]*Département de Physique, de Génie Physique, et d'Optique, Université Laval, Québec (Québec), Canada G1V 0A6*
[2]*MACSI, Department of Mathematics & Statistics, University of Limerick, Limerick, V94 T9PX, Ireland*

(Received 7 November 2017; published 2 March 2018)

In binary cascade dynamics, the nodes of a graph are in one of two possible states (inactive, active), and nodes in the inactive state make an irreversible transition to the active state, as soon as their precursors satisfy a predetermined condition. We introduce a set of recursive equations to compute the probability of reaching any final state, given an initial state, and a specification of the transition probability function of each node. Because the naive recursive approach for solving these equations takes factorial time in the number of nodes, we also introduce an accelerated algorithm, built around a breath-first search procedure. This algorithm solves the equations as efficiently as possible in exponential time.

DOI: 10.1103/PhysRevE.97.032302

I. INTRODUCTION

Irreversible binary-state dynamics model rapid information transmission in complex systems. In these dynamics, nodes are in one of two states (inactive, active), and nodes in the inactive state make an irreversible transition to the active state, as soon as their precursors satisfy a predetermined condition. In a popular formulation of the problem [1], transition conditions are expressed by a set of response functions $\{F_i(m)\}_{i=1,\ldots,N}$ that give the probabilities that nodes will make a transition from an inactive to an active state, based on the number m of their active precursors. This general formulation, known as *cascade dynamics*, encompasses multiple important dynamics as special cases [1]. Noteworthy examples are site and bond percolation [2,3], the Watts model of threshold dynamics [4], and susceptible-infected models [5]. As a result, cascade dynamics have relevant applications in fields as diverse as epidemiology [5,6], economics [7], and neuroscience [8–10].

Cascade dynamics present a difficult mathematical challenge: Predicting its outcome on arbitrary network topologies is notoriously hard. Only in some special cases do we know of analytical solutions that are both simple and elegant. Perhaps the most famous special case is that of ensembles of treelike networks [11–13], for which probability generating functions (PGF) [1,3,14–16] and message passing (MP) [17,18] methods yield exact analytical predictions of the important observables of the dynamics (e.g., size of the giant component, critical propagation threshold, etc.). In fact, their predictions are so accurate that these methods are routinely applied to real networks, even though the underlying hypotheses are no longer valid; in many cases, this leads to surprisingly good approximations of the true outcome [19,20]. However in many important cases, experiment and theory are at odds. It is now understood that these discrepancies can—in part—be traced back to existence of local correlations in real systems, whose effects are overlooked by tree-based theory [19].

There are a few promising methods—developed within the general framework of cascade dynamics—that address this issue by including local correlations using, e.g., a fixed prevalence of cliques on a treelike backbone [21,22] or with fixed degree-degree correlations [16]. However, the most versatile proposition to date comes from percolation theory [23–25]; it consists in mixing PGF methods with exact solutions on arbitrary motifs. The general idea behind this method is simple: One first decomposes a network in small local subgraphs (motifs), then solves percolation on these graphs, and, finally, combines the local solutions to obtain a global solution [24].

There appears to be no conceptual obstacles to adapting this method to general cascade dynamics—the clique-based methods of Refs. [21,22] are in fact special cases of the general approach. There is, however, a major practical bottleneck: Exactly solving cascade dynamics on small graphs is, at best, tedious [22]. This bottleneck becomes a barrier when motifs are too diverse, or too large. In Refs. [24,26], a costly but systematic algorithm is introduced to handle enumeration and averaging of the traversal probabilities in the *percolation problem*. The goal of the present paper is to delineate the equivalent enumeration algorithm in the much more general context of *cascade dynamics*.

The paper is organized as follows. We first define cascade dynamics, and obtain recursive equations for the probability of every outcome, on arbitrary network topologies and general cascades (Sec. II). We then discuss the practical aspect of the formalism in Sec. III, i.e., how to compute the solution of the recursive equations as efficiently as possible.[1] In Sec. IV, we illustrate the power of the formalism in a case study of complicated mixed dynamics occurring on small directed

*edward.laurence.1@ulaval.ca
†ljd@phy.ulaval.ca

[1]We also provide a reference implementation of our solver at github.com/laurencee9/exact_binary_dynamics.

networks. We gather our conclusions and perspectives in Sec. V. Three appendices follow. In the first, we prove that our method is equivalent to the analogous method derived in the context of percolation theory (Appendix A). In the second, we work out an explicit example (Appendix B). In the third and last Appendix, we give a detailed calculations of the worst-case computational complexity of our algorithms (Appendix C).

II. CASCADE DYNAMICS ON ARBITRARY GRAPHS

A. Definition of the dynamics

We consider an irreversible binary-state dynamics occurring on an arbitrary network of N nodes. This dynamic process is well defined on graphs that contain directed edges [1], and we therefore encode the structure of the network in a binary-valued, potentially asymmetric $N \times N$ adjacency matrix $\boldsymbol{A}$. We adopt the convention that the element a_{ij} of $\boldsymbol{A}$ indicates the existence of an edge going from node j to node i. In the directed case, the neighborhood of node i, i.e., the set of nodes that can be reached from i, is hence given by the nonzero entries on the ith column of $\boldsymbol{A}$, and the precursors of node i are the set of nodes having a directed edge to node i. The undirected case is obtained by placing a directed edge in both directions.

Following the prescription of Ref. [1], all nodes are initially placed in the inactive state, except for a (potentially disconnected) set of seed nodes, initially active. At each subsequent discrete time steps t, inactive nodes make a transition to the active state if their precursors satisfies an activation condition. The process ends when no further transitions are possible (i.e., when the state of each node at time t is identical to the state at time $t+1$).

The transition conditions are encoded in a set of N so-called "influence response functions" $\{F_i(m)\}_{i=1,..,N}$, where m is the number of active precursors of node i [1]. We define these functions as the cumulative distribution function (CDF) of the probability $P_i(m)$ that node i has a hidden activation threshold precisely equal to m precursors:

$$P_i(m) = F_i(m) - F_i(m-1), \tag{1}$$

or in other words, as the CDF of the probability that node i will make a transition to the active state, as soon as m of its precursors reach the active state. For convenience, we also define the complementary probability

$$G_i(m) := 1 - F_i(m) \tag{2}$$

that node i has a hidden threshold greater than m active precursors—or, equivalently, that it does not make a transition when it has m active precursors. Under this probabilistic description, the process ends when every inactive node at time t fails to make a transition to the active state.

A complete specification of the dynamics thus consists of the structure of the network, as specified by a $N \times N$ adjacency matrix $\boldsymbol{A}$; a set of seed nodes; and a response function $F_i(m)$ for each node. Because our formalism allows it, we will work with response functions specified on a node-to-node basis. Note, however, that it is not uncommon to group nodes in coarser compartments. In fact, response functions $F(m_i,\theta_i)$ that only depend on m_i and some local parameter θ_i—such as the degree k_i of node i—are widely used [15]: Many well-known dynamics can be recovered as a special case by choosing specific classes of response functions [1].

B. Special classes of response functions

Let us consider first bond percolation, where a node is part of an active component if at least one of its incoming edges percolates. If one denotes by m_i the number of in-edges, that can percolate, of node i, and by p the percolation probability, then node i remains inactive with probability $G(m_i,p) = (1-p)^{m_i}$. The response function can therefore be written as the complementary probability

$$F(m_i) = 1 - (1-p)^{m_i}. \tag{3a}$$

For site percolation, where a node percolates with probability p if it has at least a single active precursor, the response function is simply

$$F(m_i) = \begin{cases} 0, & m_i = 0 \\ p, & m_i > 0 \end{cases}. \tag{3b}$$

In the Watts cascade model [4], a node of degree k_i makes a transition to the active state when a critical fraction r of its k_i precursors reaches the active state, and the thresholds r are drawn from a distribution, whose cumulative distribution is given by $C(r)$. The response function for the resulting dynamics is therefore

$$F(m_i,k_i) = C(m_i/k_i). \tag{3c}$$

These are but a few examples, derived in Ref. [1], and reproduced here to showcase the generality of the cascade formalism. Furthermore, there exists many more equivalencies, inherited from its reduction to the Watts threshold model [see Eq. (3c)]; the latter can be mapped, with the appropriate choice of thresholds, to site percolation, k-core percolation, diffusion percolation [27], and the generalized epidemic process [28]. Our method therefore provides systematic and exact solutions to a large class of dynamics.

C. Exact recursive solution

The outcome of a cascade dynamics on a network is a configuration of active and inactive nodes. We encode these configurations in binary vectors $\boldsymbol{l} = [l_1,\ldots,l_N]^\intercal$ of length N, where $l_i = 1$ if node i is active in the configuration and 0 otherwise. Thus, the vector $\boldsymbol{n} := [1,\ldots,1]^\intercal$ refers to the completely active configuration, and the number of active nodes in a configuration $\boldsymbol{l}$ is given by the square of its Euclidean norm, e.g., $|\boldsymbol{n}|^2 = N$.

Cascade dynamics are, by definition, probabilistic processes. So fully solving the cascade amounts to computing the probability of every outcome. Given a fixed initial configuration $\boldsymbol{l}_0$ and a fixed structure $\boldsymbol{A}$, we define $Q(\boldsymbol{l};\boldsymbol{A},\boldsymbol{l}_0)$ as the probability of observing configuration $\boldsymbol{l}$ when the cascade stops. A solution is therefore a distribution $\mathcal{Q} = \{Q(\boldsymbol{l};\boldsymbol{A},\boldsymbol{l}_0)\}_{\boldsymbol{l}}$ over all $\boldsymbol{l} \in \{0,1\}^N$, for a choice of response functions (not explicitly denoted, for the sake of clarity). Even though there are exponentially many possible outcomes ($|\mathcal{Q}| = 2^N$), the calculation of $\mathcal{Q}$ is greatly simplified by the structure underlying the distribution.

Observe that when a cascade process ends, the nodes can be partitioned in two subsets defined by their activity status. In particular, nodes in the inactive subset must have, by definition, a hidden threshold superior to the number of their active precursors. The probability of observing any subset of inactive nodes can thus be written in terms of the complementary probabilities $\{G_i(m)\}$. Specifically, since each activation event is independent, we can write the stopping probability of the cascade in configuration $\boldsymbol{l}$ as

$$Q(\boldsymbol{l};\boldsymbol{A_u}) = Q(\boldsymbol{l};\boldsymbol{A_l})\prod_{i=1}^{N}[G_i(m_i(\boldsymbol{l}))]^{u_i(1-l_i)}, \quad \boldsymbol{u}\geqslant\boldsymbol{l}, \tag{4a}$$

where $m_i(\boldsymbol{l}) = \sum_j a_{ij}l_j$ is the number of active precursors of node i and where $\boldsymbol{u}\geqslant\boldsymbol{l}$ is an elementwise inequality $u_i \geqslant l_i \forall i$. The matrix $\boldsymbol{A_l}$ denotes the reduced adjacency matrix

$$\boldsymbol{A_l} = \boldsymbol{L}\boldsymbol{A}\,\boldsymbol{L},$$

where $\boldsymbol{L} = \mathrm{diag}(\boldsymbol{l})$ is a $N\times N$ diagonal matrix whose entries are given by the vector $\boldsymbol{l}$. The reduced adjacency matrix $\boldsymbol{A_l}$ is almost identical to $\boldsymbol{A}$, except that the values on the jth row and column are set to zero if $l_j = 0$. It encodes the structure of the subgraph induced by the set of active nodes in configuration $\boldsymbol{l}$. Thus, if $\boldsymbol{l} = \boldsymbol{n}$, then $\boldsymbol{A_n} = \boldsymbol{A}$.

With this in mind, Eq. (4a) is interpreted as follows: It states that the probability that a cascade taking place on $\boldsymbol{A_u}$ will stop at configuration $\boldsymbol{l}$ is equal to the probability $Q(\boldsymbol{l};\boldsymbol{A_l})$ of reaching a completely active subgraph induced by $\boldsymbol{l}$ times the probability $\prod_{i=1}^{N}[G_i(m_i)]^{u_i(1-l_i)}$ that the cascade does not spread to the remaining $|\boldsymbol{u}-\boldsymbol{l}|^2$ inactive nodes.

This essentially solves the problem. The distribution of outcomes $\mathcal{Q}$ can—in principle—be computed by solving the system of $2^{(N-|\boldsymbol{l}_0|^2)}$ equations defined in (4a), recursively, for configurations with increasing numbers of active nodes. However, it turns out that this system is not complete: Every completely active configuration in a reduced subgraph is associated to a noninformative Eq. (4a), because $u_i = l_i = 1\,\forall i$ yields the tautology

$$Q(\boldsymbol{l};\boldsymbol{A_l}) = Q(\boldsymbol{l};\boldsymbol{A_l}).$$

Hence, we are dealing with an underdetermined system of equations. Fortunately, the system can be completed via the normalization condition

$$Q(\boldsymbol{l};\boldsymbol{A_l}) = 1 - \sum_{\boldsymbol{u}:\boldsymbol{u}<\boldsymbol{l}} Q(\boldsymbol{u};\boldsymbol{A_l}), \tag{4b}$$

where $\boldsymbol{u}:\boldsymbol{u}<\boldsymbol{l}$ means that the sum is taken over all $\boldsymbol{u}$ respecting the elementwise inequality $u_i < l_i \forall i$. Equation (4b) states that the probability of reaching the fully active configuration $\boldsymbol{l}$ on the reduced adjacency matrix $\boldsymbol{A_l}$ is given by the complementary probability of reaching *any other state* with fewer active nodes (on $\boldsymbol{A_l}$). Of course, one also needs to give an initial condition of the form

$$Q\big(\boldsymbol{l}_0;\boldsymbol{A}_{\boldsymbol{l}_0}\big) = 1, \quad Q(\boldsymbol{l};\boldsymbol{A_l}) = 0 \quad \forall\,\boldsymbol{l}\leqslant\boldsymbol{l}_0, \tag{4c}$$

to complete the system.

Before considering the practical aspect of this formalism, we note that an analogous derivation leads to almost identical equations in the case of bond percolation [26]. In fact, we show in Appendix A that on substitution of the response function (3a) in the equations, one recovers the formalism derived in Ref. [26] for bond percolation.

III. ENUMERATION ALGORITHMS

A. General recursive solution method

In practice, and as stated above, Eqs. (4) must be solved recursively. One must feed the results of Eq. (4a) into Eq. (4b), and back into Eq. (4a), using the initial condition Eq. (4c) as a starting point. To make things more concrete, let us follow through with the first few steps of the computation.

First, we compute the probabilities $Q(\boldsymbol{l}_0;\boldsymbol{A})$ using Eq. (4a) and the initial condition Eq. (4c). Then we proceed to configurations with one more active node, i.e., configurations $\boldsymbol{l}_i$ such that $|\boldsymbol{l}_i|^2 = |\boldsymbol{l}_0|^2 + 1$. These probabilities, of the form $Q(\boldsymbol{l}_i;\boldsymbol{A})$, can be computed from Eq. (4a). But in so doing, we will need to resort to the normalization condition [Eq. (4b)]. Once the stopping probabilities of all configurations with one extra active node are computed, it is only a matter of repeating the process for all configurations with one more active node (i.e., $\boldsymbol{l}_i$ such that $|\boldsymbol{l}_i|^2 = |\boldsymbol{l}_0|^2 + 2$), progressing toward larger and larger configurations, all the way up to the complete configuration of $|\boldsymbol{n}|^2 = N$ nodes. This scheme obviously constructs the complete distribution $\mathcal{Q}$ (as well as a complete set of distributions $\mathcal{Q}'$ on every possible subgraph of $\boldsymbol{A}$). We work out an explicit example for a small tadpole graph in Appendix B.

B. Saving time: Culling impossible configurations

The method just discussed is less than optimal, precisely because it goes through every single configuration of active nodes and every single induced subgraphs. This is due to the fact that many response functions and many graphs are associated with large sets of *impossible* configurations, i.e., sets of configurations $\boldsymbol{l}$ with null probabilities $Q(\boldsymbol{l};\boldsymbol{A_u}) = 0$. It is easy to see that a configuration $\boldsymbol{l}$ is *necessarily* impossible if it contains at least one node i not connected to a seed (or a spontaneously activated node) via a path in $\boldsymbol{A_l}$: There is then simply no possible path for the cascade to have propagated to node i. Such configurations abound.

If we can find and avoid these impossible configurations more efficiently than through brute-force enumeration of $\boldsymbol{l}\in\{0,1\}^N$, then we will have accelerated the calculation of $\mathcal{Q}$ by completely ignoring large portions of its support. This can be done in most practical cases by using a breath-first search (BFS) *over the configurations* that accounts for the presence of spontaneously activated nodes [i.e., $F_i(0) > 0$] and disconnected seeds, see Fig. 1. In this modified BFS, nodes can be in one of two states: either discovered or undiscovered. We seed the search with the initial condition $\boldsymbol{l}_0$, i.e., by labeling the $|\boldsymbol{l}_0|^2$ seed nodes as discovered. We then enumerate all *neighboring* configurations containing $|\boldsymbol{l}_0|^2 + 1$ discovered nodes, i.e., all configurations with $|\boldsymbol{l}_0|^2 + 1$ discovered nodes that are either neighbor of an initial seed node or a spontaneous active node. We then repeat the process, expanding outwards. Importantly, every new configuration is constructed by taking one of the previous configurations and converting one undiscovered node,

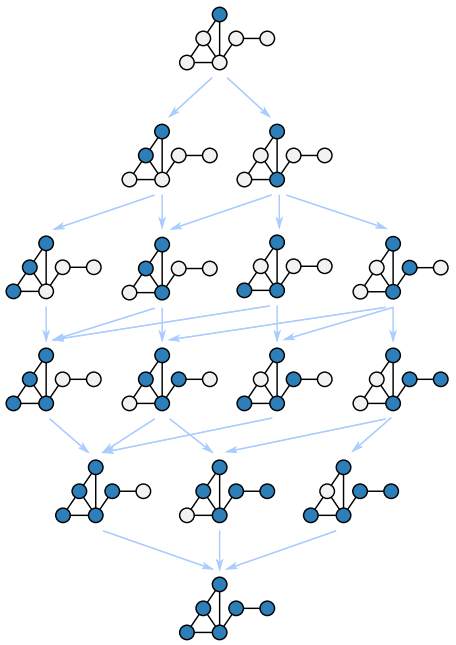

FIG. 1. Example of a breath-first exploration of all configurations. The configurations are explored by rows, i.e., by the number of discovered (active) nodes. Nodes are labeled as discovered (blue) or undiscovered (white). Blue arrows indicate possible transitions.

adjacent to either an already-discovered or a spontaneously activating node.

By carrying this process recursively, we mimic the cascade and only generate possible configurations. When there are no undiscovered nodes adjacent to a discovered node and no undiscovered spontaneous nodes, we therefore have enumerated all possible final configurations, and the algorithm terminates.

The advantage of using this search is that we can intertwine the enumeration procedure and the recursion of Eqs. (4). As soon as the number of active nodes increases by one in the BFS, say, from m to $m+1$, we know that we have encountered all possible active configurations with $m' \leqslant m$ nodes. We can thus set the probability of all the missing configurations of less than $m+1$ nodes to 0. At best, we will have saved vast amount of unnecessary calculation. At worst, the complexity of the whole calculation will remain more or less the same—the BFS will only have ordered the enumeration of configurations.

C. Saving more time: Dropping intermediary distributions

For most graphs, carrying out the full recursive solution of Eqs. (4) is impossible, because too much information must be held either in memory or because the calculation is simply too long—even with the help of the BFS strategy.

Besides the obvious exponential size of the solution, this slowdown can be traced to another important bottleneck: The evaluation of Eq. (4b) requires a complete knowledge of the distribution for all $\boldsymbol{u}$ with $\boldsymbol{u} < \boldsymbol{l}$, i.e., all $\{\mathcal{Q}'\}$ over induced subgraphs. In fact, one can show that $|\{\mathcal{Q}'\}| \sim N!$ (see Appendix. C). If we could do without the information contained in $\{\mathcal{Q}'\}$, then the limiting factor would become the (exponential) size of the solution $\mathcal{Q}$ rather the (factorial) complexity of the algorithm—an appreciable speedup. This would of course come at the price of dropping the distributions $\{\mathcal{Q}'\}$ on smaller subgraphs. One should therefore stick to the BFS enhanced recursion method of Sec. III B if this information is needed.

However, provided that $\{\mathcal{Q}'\}$ can be dropped, it is possible to compute $\mathcal{Q}$ directly. The strategy consists in combining the recursive equations and using BFS to enumerate the elements of the distribution $\mathcal{Q}$ in an orderly manner. We build on the observation that two configurations that share a common core of active nodes have related probabilities. Specifically, the probability $Q(\boldsymbol{u};\boldsymbol{A}_{\boldsymbol{l}})$ can be computed from the probability $Q(\boldsymbol{u};\boldsymbol{A})$ by considering the contribution of the inactive nodes absent from the subgraph $\boldsymbol{A}_{\boldsymbol{l}}$:

$$Q(\boldsymbol{u};\boldsymbol{A}_{\boldsymbol{l}}) = Q(\boldsymbol{u};\boldsymbol{A})\left[\prod_{i=1}^{N} G_i(m_i(\boldsymbol{u}))^{(1-l_i)(1-u_i)}\right]^{-1}. \tag{5a}$$

[we merely "factor out" complementary probabilities hidden in $Q(\boldsymbol{u};\boldsymbol{A})$.] Inserting Eq. (5a) into Eq. (4), we can then write the elements of $\mathcal{Q}$ directly as

$$Q(\boldsymbol{l};\boldsymbol{A}) = [1 - Z(\boldsymbol{l},\boldsymbol{A})]\prod_{i=1}^{N}[G_i(m_i(\boldsymbol{l}))]^{(1-l_i)}, \tag{5b}$$

where

$$Z(\boldsymbol{l},\boldsymbol{A}) := \sum_{\boldsymbol{u}:\boldsymbol{u}<\boldsymbol{l}} \frac{Q(\boldsymbol{u};\boldsymbol{A})}{\prod_{i=1}^{N} G_i(m_i(\boldsymbol{u}))^{(1-l_i)(1-u_i)}}. \tag{5c}$$

This transformation is useful because Eqs. (5b) and (5c) explicitly include the normalization condition and can be solved without recursion. The procedure is simple. We start by listing and ordering all possible configurations of the complete graph using BFS. It is important to keep the discovery order. Then we solve Eq. (5b) from the smallest configuration to the largest configuration (i.e., the discovery order). At each evaluation, every $Q(\boldsymbol{u};\boldsymbol{A})$ appearing in Eq. (5c) is already calculated and memorized—as imposed by the discovery order. Under this systematic method, the previously calculated information is reinjected in the set of equations. All intermediate distributions are thus simply dropped, which considerably reduces the computational complexity of the algorithm (see Appendix C).

IV. RESULTS AND APPLICATIONS

Our main motivation in deriving recursive equations is to elaborate methods that combines exactly solved motifs on a treelike backbone, in the spirit of Refs. [24,25]. However, for small graphs, the method can also be useful in itself. For instance, one can marginalize $\mathcal{Q}$ to obtain the individual activation probabilities of each node, and these probabilities can then be used for diagnosis purposes, e.g., the importance of a node with regard to spreading [29]. With this in mind, we give two practical examples in the next section to illustrate the power of exact solutions on small graphs.

A. Calibration: Special cases on a directed graph

To calibrate the method and verify its validity, we first use the algorithm of Sec. III C to obtain cascade results on a known network: The directed network of 20 nodes and 54 directed

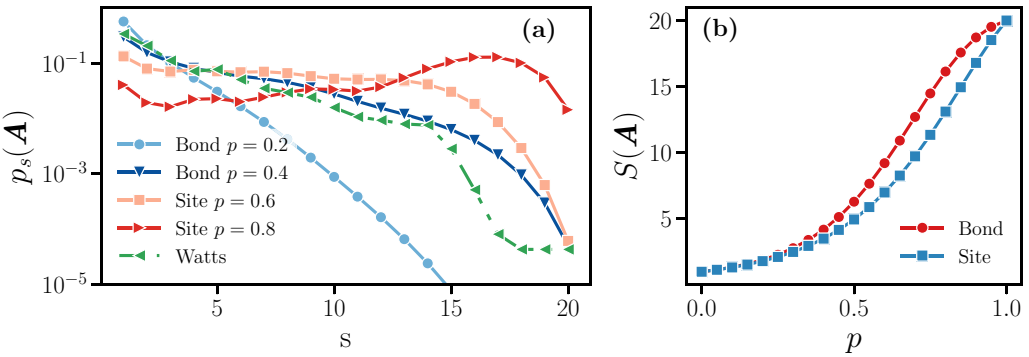

FIG. 2. (a) Size distribution of the active components under different dynamic processes: bond percolation, site percolation, and the Watts threshold model. Symbols are the results of 10^7 Monte Carlo simulations, and curves show the exact probability given by our approach [Eq. (6)]. (b) Exact mean size [Eq. (7)] of the final active component for bond and site percolations as a function of the occupation probability p. The graph has $N = 20$ and 54 directed edges.

edges appearing in Ref. [26]. We compute the solutions for three special cases of cascade dynamics, previously introduced in Sec. II B: bond percolation [see Eq. (3a)], site percolation [see Eq. (3b)], and the Watts cascade model [see Eq. (3c)]. Furthermore, we compare the predictions of the formalism with Monte Carlo simulations.

For site (bond) percolation, the Monte Carlo simulations are done by randomly selecting a seed node and then occupying the neighbors (adjacent edges) with a probability p. This step is then repeated for the new neighborhood (adjacent edges) until the process stops or until the network is exhausted. For the Watts threshold model, the simulation follows the standard description of a cascade process [4]. We first assign a threshold C_i between 0 and 1 to each node i, drawn from a uniform distribution. A randomly chosen node—the seed—is then activated. We then enter the propagation phase: If the threshold C_i of node i is lower or equal to the fraction of its precursors that are active m_i/k_i, then the node makes a transition to the active state. This process is repeated until no transitions are possible. For all these Monte Carlo simulations, the estimator $\hat{Q}(\boldsymbol{l};\boldsymbol{A},\boldsymbol{l}_0)$ of the outcome probability is computed as the frequency of the final configuration $\boldsymbol{l}$; standard results tell us that the variance of $\hat{Q}$ will decrease as the square root of the number of trials.

Note that because we seed Monte Carlo simulations at random, we compare against results computed using exact probabilities $\mathcal{Q}$ that are marginalized over every initial configuration $|\boldsymbol{l}_0|^2 = 1$, i.e.,

$$Q(\boldsymbol{l};\boldsymbol{A}) = \sum_{\boldsymbol{l}_0:|\boldsymbol{l}_0|^2=1} Q(\boldsymbol{l};\boldsymbol{A},\boldsymbol{l}_0).$$

This is consistent with the typical way in which Monte Carlo simulations are carried out.

The graph contains 20 nodes, meaning that there are roughly 10^6 different configurations—we cannot possibly visualize the complete distribution $\mathcal{Q}$. Thus, we will focus on summary statistics, the size distribution [Fig. 2(a)], and the mean size [Fig. 2(b)]. Using the distribution $\mathcal{Q}$, we calculate the probability p_s that a cascade will reach s nodes as

$$p_s(\boldsymbol{A}) = \sum_{\boldsymbol{l}\in\{0,1\}^N} Q(\boldsymbol{l};\boldsymbol{A})\mathbb{I}(|\boldsymbol{l}|^2 = s), \tag{6}$$

where $\mathbb{I}(\cdot)$ is an indicator function equal to 1 when its argument is true and 0 otherwise. We also compute the mean size of the active component,

$$S(\boldsymbol{A}) = \sum_{s=1}^{N} s p_s(\boldsymbol{A}) = \sum_{\{\boldsymbol{l}\}} Q(\boldsymbol{l};\boldsymbol{A})|\boldsymbol{l}|^2. \tag{7}$$

Figure 2(a) shows the size distribution for different dynamics and occupation probabilities. The perfect fit of the exact results and the Monte Carlo simulations confirms the validity of the equations and the algorithm. In the case of the selected graph (see Ref. [26]), roughly half of the $2^{20} = 1\,048\,576$ possible configurations go into the calculations of the size distributions. Notably, computing these distributions is trivial once the recursive procedure of Sec. III has been carried out: The equations yield a large polynomial in $\{G_i(x)\}$, which can be evaluated easily and many times by specifying the response functions and parameters. Thus, generating a family of distributions is not significantly more costly than generating a single one. Another important aspect, also raised in Ref. [26] is the irregularities of the size distributions (in the case of bond percolation), which indicates that the solution is nontrivial and depends on the intricacies of the structure of the graph. This tells us that any close form solution will necessarily be just as complex. Furthermore, we see in Fig. 2(a) that bond percolation behaves, in fact, quite simply in comparison with the other dynamics (site percolation, Watts model); exact solutions on small graphs therefore appear even more useful in the case of general cascade dynamics than in the special case of bond percolation.

Figure 2(b) shows the theoretical mean size component for two dynamics. Again, we benefit from the fact that the corresponding polynomial can be evaluated at will once the recursive procedure has been completed: Our approach allows us to zoom-in on any point of the curve and to investigate specific regions of the parameter space at little extra computational costs.

B. Mixed dynamics

In the previous case study, the network had a global response function, independent of the node identity. In real situations, it is fair to assume that nodes of different types do not respond

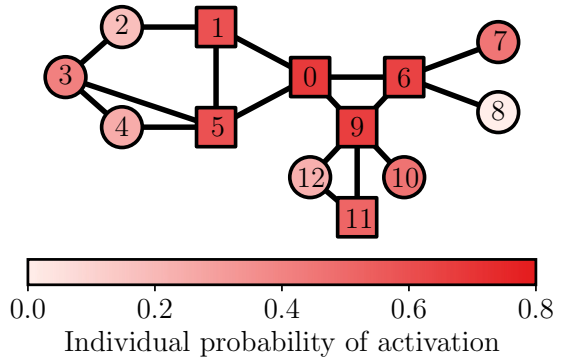

FIG. 3. Individual response function and activation probabilities. The colors of the nodes are used to show their activation probability, summed over all possible outcomes. Square nodes can make a spontaneous transition [with response function $F_s(p,q)$, see Eq. (9)], while round nodes need active precursors to make a transition [they have the response function $F_w(p,\tau)$, see Eq. (8)]. The parameters of the spontaneously activating nodes are $p = 0.4$ and $q = 0.3$ for nodes 1, 5, and 11 and $p = 0.6$ and $q = 0.1$ for nodes 0, 6, and 9. The parameters of the threshold nodes are $p = 0.6$ and $\tau = 2$ for node 2, 4, 8, and 12 and $p = 0.7$ and $\tau = 1$ for nodes 3, 7, and 10.

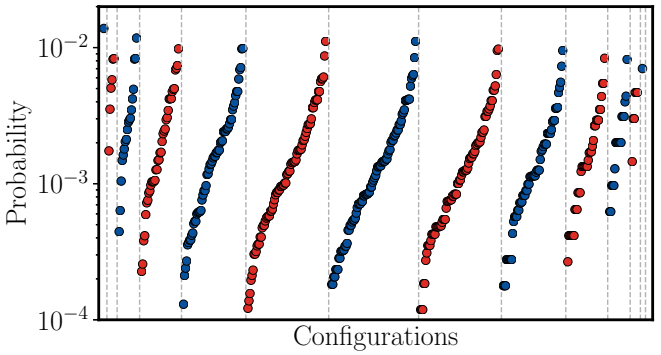

FIG. 4. Complete distribution $\mathcal{Q}$ for the every possible configurations of the network, using the dynamics described in Fig. 3. Configurations are grouped together according to the number of active nodes (from 0 to 12) and then ordered with ascending probabilities. Since node 8 is always inactive, the completely active configuration (13 active nodes) has a zero probability and, consequently, is not shown. Gray lines are placed where the number of active nodes increases.

identically to stimuli. Thus, it is natural and perfectly general to consider a mixed dynamics that stems from diverse response functions. As highlighted in Sec. II, our formalism can handle this generalization without any additional complexity.

We consider a mixed dynamics where nodes are associated with one of two types of parameterizable response function,

$$F_w(p,\tau) = \begin{cases} 0 & m \leqslant \tau - 1 \\ p & m \geqslant \tau \end{cases}, \tag{8}$$

$$F_s(p,q) = \begin{cases} p & m = 0 \\ 1 - (1-p)q^m & m > 0 \end{cases}. \tag{9}$$

The function $F_w(p,\tau)$ describes a threshold dynamics, where τ is the activation threshold and p is the activation probability once the threshold is exceeded. The function $F_s(p,q)$ describes a node that can either activate spontaneously, or through contagion; p is the probability that the node will spontaneously activate (at time $t = 0$), whereas q controls its sensitivity to active precursors. We assign these response functions to the $N = 13$ nodes of a handmade small network, shown in Fig. 3. The network is constructed to showcase heterogeneous patterns of activation.

Because we have introduced random seeds, we will no longer sum the probabilities over all seeds; instead, we always begin the process in the inactive configuration $\boldsymbol{l_0} = [0,0,\ldots,0]_\intercal$ and let spontaneous activations determine the outcome. In Fig. 4, the probability $Q(\boldsymbol{l};\boldsymbol{A})$ is displayed for the complete set of possible outcomes $\boldsymbol{l}$. Since the network is relatively sparse and small ($N = 13$), these results can be computed extremely quickly (less than a minute on a modern personal computer).

In Fig. 3, we have colored nodes with their probability of being active by summing over all possible outcomes. Combining the information given by both of these graphs, we see that the fully active configuration is impossible, because node 8 has a degree smaller than its threshold. Similar and much more in depth analyses could obviously be carried out on real small networks, e.g., on a power grid.

V. DISCUSSION

In this paper, we have derived a new set of recursive equations that solve cascade dynamics on arbitrary networks, and we have introduced two practical strategies to manage the complexity of solving such a large set of equations. These developments are inspired by analogous methods, introduced within the framework of percolation theory [26].

Due to the generality of the cascade dynamics formulation [1,28], our method leads to exact solutions for a wide range of dynamics, including well-known examples such as site and bond percolation and the Watts threshold model. But its power goes beyond such simple cases; it also generates exact solution for many exotic dynamics and connectivity patterns, e.g., directed, self-referencing edges, weighted graphs, disconnected active components, spontaneous activity and disconnected initial seeds activation. Furthermore, the exact solution is valid in the much more general context of individual activation functions (i.e., each node can have a different activation function). To the best of our knowledge, our framework is the only one able to do so.

We cannot understate the fact that our formalism solves a problem whose solution grows exponentially with N. As such, the method is, by necessity, intractable.[2] On the other hand, the usual methods for solving numerically cascade dynamics, the tree-based theory and the message-passing framework [18,20,30], are surprisingly accurate for sparse networks and are commonly used on real large networks [19]. Our contribution is therefore not designed for computing ensemble statistics in the large-N limit, where Monte Carlo simulations

[2]In the worst-case scenario of a complete graph, our implementation is able to handle roughly 20 nodes on a modern single-CPU computer.

or approximations schemes would be more appropriate if the specifics of the configurations do not matter. Instead, our method is designed to yield exact solutions for small graphs, where the exponential dependency is still manageable. This is useful in at least three scenarios: (i) on small graphs where accurate *configuration* probabilities are needed (cf. Ref. [29]), (ii) in formalisms where motif distributions are specified and traversal probabilities must preferably be computed in closed form (cf. Refs. [23,24,26]), and (iii) on graphs with mixed dynamics. In all cases, our formalism involves calculations that are no more complex than Monte Carlo simulations, with the added advantage of producing exact probabilities as well.

In summary, despite the unwieldiness of the calculations involved, our results open the way to new theoretical predictions, since it solves cascades on small motifs, both exactly and systematically.

ACKNOWLEDGMENTS

We are thankful to Antoine Allard and Patrick Desrosiers for useful comments and suggestions. We are grateful to Andrey Lokhov for pointing out Ref. [20]. This work was funded by the Fonds de recherche du Québec-Nature et technologies (J.-G.Y. and E.L.), the Conseil de recherches en sciences naturelles et en génie du Canada (L.J.D.), the Irish Research Council (New Foundations grant to S.M.), Science Foundation Ireland (Grant No. 11/PI/1026, S.M.), and Sentinel North (E.L., J.-G.Y). E.L. and J.-G.Y. contributed equally to this work.

APPENDIX A: EQUIVALENCE WITH BOND PERCOLATION

An exact solution to bond percolation is given in Ref. [26], in the form of a set of recursive equations. We show that our equations generalize these equations. As stated in Sec. II B, bond percolation can be mapped to a cascade dynamic process by setting $F(m_i) = 1 - (1-p)^{m_i}$ and $G(m_i) = (1-p)^{m_i}$ for all nodes. We insert these relations in Eqs. (4a) and obtain

$$Q(\boldsymbol{l}; \boldsymbol{A_u}) = Q(\boldsymbol{l}; \boldsymbol{A_l}) \prod_{i=1}^{N} [(1-p)^{m_i}]^{u_i - l_i} \qquad \boldsymbol{u} \geqslant \boldsymbol{l}. \tag{A1}$$

Next, writing out the number of active precursors of i as $m_i(\boldsymbol{l}) = \sum_k a_{ik} l_k$, we get

$$\prod_{i=1}^{N} (1-p)^{\sum_k a_{ik} l_k \bar{l}_i} = (1-p)^{\sum_{k,i} a_{ik} l_k \bar{l}_i} ,$$

$$= (1-p)^{\boldsymbol{l}^T \boldsymbol{A} \bar{\boldsymbol{l}}} ,$$

where $\bar{l}_i := u_i - l_i$. Substituting back into Eq. (A1), we find, for $\boldsymbol{u} = \boldsymbol{n}$,

$$Q(\boldsymbol{l}; \boldsymbol{A}) = Q(\boldsymbol{l}; \boldsymbol{A_l})(1-p)^{\boldsymbol{l}^T \boldsymbol{A} \bar{\boldsymbol{l}}}, \tag{A2}$$

i.e., Eq. (3) of Ref. [26]. The normalization condition is simply generalized since it does not depend on the specifics of the cascade dynamics. This completes the proof of equivalence.

APPENDIX B: EXPLICIT EXAMPLE ON A TADPOLE GRAPH

In this Appendix, we work out several of the first steps of the full recursive procedure on a small tadpole graph of four nodes, i.e., the graph whose adjacency matrix is given by

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{B1}$$

We choose an initial configuration where every node is inactive, $\boldsymbol{l_0} = [0,0,0,0]^\intercal$. To keep the response function as general as possible, we will not further specify the dynamics. Therefore, spontaneous activation $[F_i(m) > 0]$ is assumed to occur; otherwise, the only possible outcome is $\boldsymbol{l_0}$. To simplify the demonstration, we suppose identical response functions for each node, i.e., $F_i(m) = F(m_i) \forall i$.

There exists 2^4 different configurations of the cascade dynamics. We start with the initial configuration and Eq. (4a) to yield

$$Q(\boldsymbol{l_0}; \boldsymbol{A}) = Q(\boldsymbol{l_0}; \boldsymbol{A_{l_0}}) G(0)^4 = G(0)^4,$$

since $Q(\boldsymbol{l_0}; \boldsymbol{A_{l_0}}) = 1$ by definition [see the initial condition in Eq. (4c)]. Moving to configurations with one more active node, we begin with $\boldsymbol{l_1} = [1,0,0,0]^\intercal$ and follow the same procedure,

$$\begin{aligned} Q(\boldsymbol{l_1}; \boldsymbol{A}) &= Q(\boldsymbol{l_1}; \boldsymbol{A_{l_1}}) G(1)^2 G(0) \\ &= [1 - Q(\boldsymbol{l_0}; \boldsymbol{A_{l_1}})] G(1)^2 G(0) \\ &= [1 - Q(\boldsymbol{l_0}; \boldsymbol{A_{l_0}}) G(0)] G(1)^2 G(0) \\ &= F(0) G(1)^2 G(0), \end{aligned} \tag{B2}$$

where we have used the definition $F(m) = 1 - G(m)$ at the last step. Notice how the normalization (4c) intervenes and how $Q(\boldsymbol{l_0}; \boldsymbol{A_{l_1}})$ and $Q(\boldsymbol{l_1}; \boldsymbol{A_{l_1}})$ are computed as a by-product of this step, producing the complete distribution of outcomes $\mathcal{Q}'$ for a cascade taking place on $\boldsymbol{A_{l_1}}$. Further steps will generate similar distributions.

Completing our calculations for the other 1 node configurations, we observe that the symmetry of the graph leads to $Q(\boldsymbol{l_2}; \boldsymbol{A}) = Q(\boldsymbol{l_1}; \boldsymbol{A}) = F(0) G(1)^2 G(0)$ with $\boldsymbol{l_2} = [0,1,0,0]^\intercal$, while

$$Q(\boldsymbol{l_3} = [0,0,1,0]^\intercal; \boldsymbol{A}) = F(0) G(1)^3 ,$$

$$Q(\boldsymbol{l_4} = [0,0,0,1]^\intercal; \boldsymbol{A}) = F(0) G(1) G(0)^2 ,$$

following the procedure of Eq. (B2).

For configurations with two active nodes such as $\boldsymbol{l_5} = [1,1,0,0]^\intercal$, two steps of recursions are required. First, we use Eq. (4a)

$$Q(\boldsymbol{l_5}; \boldsymbol{A}) = Q(\boldsymbol{l_5}; \boldsymbol{A_{l_5}}) G(2) G(0)$$

and apply Eq. (4b)

$$Q(\boldsymbol{l_5}; \boldsymbol{A_{l_5}}) = [1 - Q(\boldsymbol{l_0}; \boldsymbol{A_{l_5}}) - Q(\boldsymbol{l_1}; \boldsymbol{A_{l_5}}) - Q(\boldsymbol{l_2}; \boldsymbol{A_{l_5}})].$$

None of these terms are *a priori* known. We must use Eq. (4a) again for each of them

$$Q(\boldsymbol{l_0};\boldsymbol{A}_{l_5}) = Q(\boldsymbol{l_0};\boldsymbol{A}_{l_0})G(0)^2 = G(0)^2,$$
$$Q(\boldsymbol{l_1};\boldsymbol{A}_{l_5}) = Q(\boldsymbol{l_1};\boldsymbol{A}_{l_1})G(1) = F(1)G(1),$$
$$Q(\boldsymbol{l_2};\boldsymbol{A}_{l_5}) = Q(\boldsymbol{l_2};\boldsymbol{A}_{l_2})G(1) = F(1)G(1).$$

This leads to

$$Q(\boldsymbol{l_5};\boldsymbol{A}) = [1 - G(0)^2 - 2F(1)G(1)]G(2)G(0).$$

This process can obviously be carried out systematically for the five other configurations with two active nodes. We then proceed to larger configurations until we reach $\boldsymbol{l} = \boldsymbol{n}$, which requires a special treatment. Rather than using Eq. (4a), we directly use the normalization (4b) to find $Q(\boldsymbol{n};\boldsymbol{A})$. This completes the calculation of $\mathcal{Q}$ on $\boldsymbol{A}$, and all other distributions $\mathcal{Q}' < \mathcal{Q}$ have been computed in the process.

APPENDIX C: COMPLEXITY CALCULATION

In the main text, we mention that the exact recursive solution of Eqs. (4) is much more computationally complex than the accelerated solution, found using Eqs. (5b), since the latter skips the computation of $\mathcal{Q}'$ on the induced subgraphs. This Appendix clarifies and quantifies this statement. We consider the worst case: An undirected complete graph of N nodes, with an undefined dynamics and no initial activation, i.e., $|\boldsymbol{l_0}|^2 = 0$. In this setup, the number of possible configurations is 2^N and the BFS is unnecessary. We assume that the complexity of the process is well represented by the number of $Q(\boldsymbol{l};\boldsymbol{A_u})$ required to obtain a solution.

We begin with the analysis of the complexity of the algorithm of Sec. III C (the accelerated solution). For a configuration $\boldsymbol{l}$, we count the number of terms $\mathcal{N}_{|\boldsymbol{l}|^2}$ involved in the evaluation of $Q(\boldsymbol{l};\boldsymbol{A})$ from Eq. (5b),

$$\mathcal{N}_{|\boldsymbol{l}|^2} = O\left[\sum_{|\boldsymbol{u}|<|\boldsymbol{l}|} Q(\boldsymbol{u};\boldsymbol{A}_{\boldsymbol{n}})\right]. \tag{C1}$$

For each $|\boldsymbol{u}|^2$ of the sum, there exists $\binom{|\boldsymbol{l}|^2}{|\boldsymbol{u}|^2}$ different configurations, meaning that

$$\mathcal{N}_{|\boldsymbol{l}|^2} = \sum_{|\boldsymbol{u}|^2=0}^{|\boldsymbol{l}|^2-1} \binom{|\boldsymbol{l}|^2}{|\boldsymbol{u}|^2} \sim 2^{|\boldsymbol{l}|^2}. \tag{C2}$$

For a configuration $\boldsymbol{l}$, roughly $2^{|\boldsymbol{l}|^2}$ terms are needed to obtain a solution to Eq. (5b). To obtain the complexity of the complete distribution $\{Q(\boldsymbol{l};\boldsymbol{A})\}_{\boldsymbol{l}}$, we sum up the complexity of all 2^N equations,

$$\mathcal{N}_{\text{total}} = \sum_{\boldsymbol{l}} \mathcal{N}_{|\boldsymbol{l}|^2}, \tag{C3}$$

$$= \sum_{|\boldsymbol{l}|^2=0}^{N} \binom{N}{|\boldsymbol{l}|^2} 2^{|\boldsymbol{l}|^2} = 3^N. \tag{C4}$$

We conclude that the total number of operations scales approximately as 3^N for a complete set of configurations for an arbitrary dynamics on a complete graph, using the algorithm of Sec. III C.

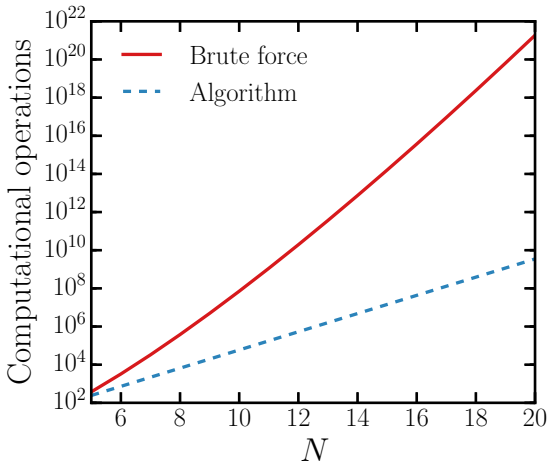

FIG. 5. Worst-case complexity of a brute-force solution of the Eqs. (4), and of the accelerated solution Eq. (5b). The complexity is assumed to be proportional to the number of $Q(\boldsymbol{l};\boldsymbol{A_u})$ required to obtain a solution. The worst case corresponds to an arbitrary dynamics with spontaneous activation on a complete graph of N nodes.

Next, we analyze the complexity of the naive recursion algorithm (see Sec. III A) in the same manner. We call this method the brute-force method, since it does not use any shortcuts. Again, the complexity is represented by the number of $Q(\boldsymbol{l};\boldsymbol{A_u})$ required to obtain a solution. The dominant contribution to the complexity comes from Eq. (4b), i.e., the normalization which needs to be evaluated for every $\mathcal{Q}'$ on all the induced subgraphs. We denote by $\mathcal{M}_{|\boldsymbol{l}|^2}$ the complexity of the brute-force calculation of $Q(\boldsymbol{l};\boldsymbol{A_u})$. The complexity can be written as a recursive equation:

$$\mathcal{M}_{|\boldsymbol{l}|^2} = O\left[\sum_{|\boldsymbol{u}|<|\boldsymbol{l}|} Q(\boldsymbol{u};\boldsymbol{A}_{\boldsymbol{l}})\right], \tag{C5}$$

$$\mathcal{M}_{|\boldsymbol{l}|^2} = \sum_{|\boldsymbol{u}|^2=0}^{|\boldsymbol{l}|^2-1} \mathcal{M}_{|\boldsymbol{u}|^2} \binom{|\boldsymbol{l}|^2}{|\boldsymbol{u}|^2}, \tag{C6}$$

since the normalization must be invoked all the way down to the initial condition. Finally, summing $\mathcal{M}_{|\boldsymbol{l}|^2}$ over each configuration to obtain the total complexity for a complete set of outcomes, we find

$$\mathcal{M}_{\text{total}} = \sum_{|\boldsymbol{l}|^2=0}^{N} \binom{N}{|\boldsymbol{l}|^2} \mathcal{M}_{|\boldsymbol{l}|^2}. \tag{C7}$$

For large $|\boldsymbol{l}|^2$, the last element of the series, i.e., $|\boldsymbol{u}|^2 = |\boldsymbol{l}|^2 - 1$, is dominant and $\mathcal{M}_{|\boldsymbol{l}|^2} \sim |\boldsymbol{l}|^2!$ [see Eq. (C6)]. Thus, $\mathcal{M}_{\text{total}} \sim N!$ for large N.

Figure 5 compares Eqs. (C4) and (C7) for different graph sizes. It is clear that the brute-force method becomes impractically *much faster* than the accelerated method.

[1] J. P. Gleeson, Phys. Rev. E **77**, 046117 (2008).
[2] S. R. Broadbent and J. M. Hammersley, in *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 53 (Cambridge University Press, Cambridge, UK, 1957), pp. 629–641.
[3] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts, Phys. Rev. Lett. **85**, 5468 (2000).
[4] D. J. Watts, Proc. Natl. Acad. Sci. USA **99**, 5766 (2002).
[5] M. E. J. Newman, Phys. Rev. E **66**, 016128 (2002).
[6] L. Meyers, Bull. Amer. Math. Soc. **44**, 63 (2007).
[7] A. G. Haldane and R. M. May, Nature **469**, 351 (2011).
[8] D. W. Zhou, D. D. Mowrey, P. Tang, and Y. Xu, Phys. Rev. Lett. **115**, 108103 (2015).
[9] M. Kaiser, M. Görner, and C. C. Hilgetag, New J. Phys. **9**, 110 (2007).
[10] M. Kaiser and C. C. Hilgetag, Front. Neuroinform. **4**, 1662 (2010).
[11] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, Phys. Rev. E. **64**, 026118 (2001).
[12] M. Molloy and B. Reed, Random Struc. Algor. **6**, 161 (1995).
[13] M. Molloy and B. Reed, Comb. Probab. Comput. **7**, 295 (1998).
[14] H. Wilf, *Generatingfunctionology* (Academic Press, New York, 1990).
[15] J. P. Gleeson, Phys. Rev. X **3**, 021004 (2013).
[16] S. Melnik, M. A. Porter, P. J. Mucha, and J. P. Gleeson, Chaos **24**, 023106 (2014).
[17] M. Shrestha and C. Moore, Phys. Rev. E **89**, 022805 (2014).
[18] B. Karrer, M. E. J. Newman, and L. Zdeborová, Phys. Rev. Lett. **113**, 208702 (2014).
[19] S. Melnik, A. Hackett, M. A. Porter, P. J. Mucha, and J. P. Gleeson, Phys. Rev. E **83**, 036112 (2011).
[20] A. Y. Lokhov, M. Mézard, and L. Zdeborová, Phys. Rev. E **91**, 012811 (2015).
[21] A. Hackett, S. Melnik, and J. P. Gleeson, Phys. Rev. E **83**, 056107 (2011).
[22] A. Hackett and J. P. Gleeson, Phys. Rev. E **87**, 062801 (2013).
[23] A. Allard, L. Hébert-Dufresne, P.-A. Noël, V. Marceau, and L. J. Dubé, J. Phys. A **45**, 405005 (2012).
[24] A. Allard, L. Hébert-Dufresne, J.-G. Young, and L. J. Dubé, Phys. Rev. E. **92**, 062807 (2015).
[25] B. Karrer and M. E. J. Newman, Phys. Rev. E. **82**, 066118 (2010).
[26] A. Allard, L. Hébert-Dufresne, P.-A. Noël, V. Marceau, and L. J. Dubé, Europhys. Lett. **98**, 16001 (2012).
[27] J. Adler and A. Aharony, J. Phys. A: Math. Gen. **21**, 1387 (1988).
[28] J. C. Miller, Phys. Rev. E **94**, 032313 (2016).
[29] P. Holme, Phys. Rev. E **96**, 062305 (2017).
[30] J. P. Gleeson and M. A. Porter, arXiv:1703.08046 (2017).