IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 February 2024, accepted 4 March 2024, date of publication 7 March 2024, date of current version 13 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3374379

RESEARCH ARTICLE

A Semi-Analytic Algorithm to Estimate Clusters With Loops in Percolation on Real Networks

CHENGUANG LI[1], TAO FU[2], LIAN WANG[2], AND RAN SUN[3]
[1]Economics and Management School, North China University of Technology, Beijing 100144, China
[2]Economics and Management School, Beijing University of Technology, Beijing 100124, China
[3]Computer School, North China Institute of Aerospace Engineering, Langfang 065000, China

Corresponding author: Tao Fu (futao@bjut.edu.cn)

This work was supported in part by Beijing Natural Science Foundation under Grant 9222010, and in part by the Humanities and Social Science Foundation of the Ministry of Education of the People's Republic of China under Grant 20yjczh066.

ABSTRACT Estimating the percolating cluster fraction is central to many percolation models. For real networks, the total size of clusters with loops can be considered a plausible metric for this fraction. In this paper, we develop a semi-analytic algorithm to estimate clusters with loops for both site and bond percolation via modifying the message passing algorithm. We compared the estimates of the original message passing algorithm and our modified version with simulation results on four real networks. Our findings suggest that our modified algorithm can achieve accuracy for any real network, provided that a sufficient number of possible states following site or bond occupation are selected and analyzed to calculate the final estimate.

INDEX TERMS Percolation, real network, message passing algorithm, pseudo-random generation.

I. INTRODUCTION

Percolation discusses the connectivity of lattices or networks under the same site or bond occupation probability, and it is a well studied topic in statistical physics. When the occupation probability is reaching some threshold, the percolation phase transition will take place, and a percolating cluster begins to appear at that moment. Estimating the fraction of the percolating cluster when the occupation probability is above the percolation threshold is a fundamental component of many percolation analytic models or algorithms (e.g. [1], [2], [3], [4]). Among them, the message passing algorithm proposed by Karrer and Newman [5], [6] is pretty outstanding and discussed or further applied by many following papers (e.g. [7], [8], [9], [10], [11]). This algorithm will produce two self-consistent equations for each edge of an undirected network. Those equations could be solved by numerical iterations, and their solutions will be used to calculate the fraction of the percolating cluster. By this manner, the message passing algorithm actually takes the details of local network topology into consideration, which makes its estimates more accurate than other percolation models or algorithms. This paper aims to develop a modified algorithm at the base of the message passing algorithm, inheriting its advantages and improving its defects.

The essence of the message passing algorithm is to estimate the overall size of all clusters with loops. As in an infinite network, the probability for a loop existing in a non-percolating cluster could be negligible [12], to calculate the total size of clusters with loops as an estimate of the percolating cluster would make sense. However, networks in the real world usually have finite nodes. In a real network, if we assume that clusters with loops are parts of the percolating cluster, we actually consider that in an infinite network corresponding to the present network, those clusters with loops belong to the percolating cluster. If we further measure the percolating cluster in numerical simulations based on the total size of all clusters with loops, instead of the size of the largest existing cluster, the message passing algorithm will yield highly accurate estimates on most networks.

According to the proposers of the message passing algorithm [5], it performs good when the number of short loops in the network is small. Meanwhile, they also emphasized that for some networks with many short loops, its estimates are still accurate. Upon further observation, we have found that this algorithm does not perform well when the

The associate editor coordinating the review of this manuscript and approving it for publication was Feiqi Deng.

network mainly consists of short loops. Additionally, for a network containing a relatively large number of short loops, the accuracy of this algorithm is uncertain. On some such networks, the algorithm may produce accurate estimates, while on others it may not do so. Nevertheless, it will be of significance to modify this algorithm to make it applicable to any network.

We start by demonstrating the message passing algorithm briefly, followed by an analysis of an unsuccessful estimation case by the algorithm to identify the source of inaccuracy. Using these insights, we develop a modified algorithm for both site and bond percolation. We compare its estimates with those of the original algorithm and numerical simulations on four real networks. Since our modified algorithm is semi-analytic, we also discuss the impacts of pseudo-random generation and the relationship between its accuracy and the corresponding calculation amount.

II. A SEMI-ANALYTIC ALGORITHM TO ESTIMATE CLUSTERS WITH LOOPS

A. A DEMONSTRATION OF THE ORIGINAL MESSAGE PASSING ALGORITHM

The message passing algorithm has already been applied to bond percolation [5] and site percolation [13] on undirected networks, respectively. Here, we would like to present simple demonstrations for both cases. Let us choose node j as the focus of observation, and N_j denotes the set of its direct neighbors. Let node i be a direct neighbor of node j, and node k is another direct neighbor of node j, and then we have $i \in N_j$ and $k \in N_j \backslash i$, where $N_j \backslash i$ denotes the set of direct neighbors of j excluding i.

For site percolation, we use $u_{i,j}$ to denote the probability that the edge linking nodes i and j in the direction i -> j(from here on, it will be referred as edge (i -> j)) is not part of the percolating cluster after the site occupation. Let p denote the site occupation probability, and we will obtain

$$u_{i,j} = 1 - p + p \cdot \prod_{k \in N_j \backslash i} u_{j,k}. \tag{1}$$

The reason behind Eq. (1) is very straightforward. If node j is not occupied (this probability is 1-p), edge (i -> j) will also not belong to the percolating cluster. Otherwise, the condition for edge (i -> j) to not be part of the percolating cluster is that all edges emanating from node j and pointing towards its neighbors, excluding node i, must also not be part of the percolating cluster.

In this way, we can list two self-consistent equations for each edge, and then solve all those equations by numerical iterations. From Eq. (1) we can see that $u_{i,j} = 1$ for all i, j is always a set of solutions. If p is higher than the percolation threshold p_c, another set of solutions will exist. The percolation threshold p_c could be given by the inverse of the leading eigenvalue of the non-backtracking matrix [14], [15], [16]. Then let S_i denote the probability that node i belongs to the

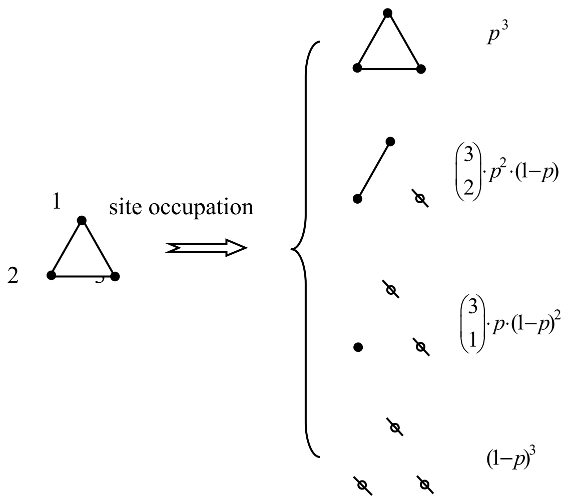

1
2
3
site occupation
p^3
$\binom{3}{2} \cdot p^2 \cdot (1-p)$
$\binom{3}{1} \cdot p \cdot (1-p)^2$
$(1-p)^3$

FIGURE 1. A diagramatical representation of site occupation on an undirected triangle, where p denotes the site occupation probability.

percolating cluster, and we have

$$S_i = p \cdot (1 - \prod_{j \in N_i} u_{i,j}). \tag{2}$$

By substituting the solutions of those self-consistent equations represented by Eq. (1) into Eq. (2), S_i could be calculated. Finally, we can calculate the percolating cluster fraction S by averaging all S_i:

$$S = \frac{\sum_{i=1}^{n} S_i}{n}, \tag{3}$$

where n is the total number of nodes.

For bond percolation, we have the same self-consistent equation as Eq.(1), where p denotes the bond occupation probability and $u_{i,j}$(or $u_{j,k}$) denotes the probability that edge (i -> j) (or edge (j -> k)) is not part of the percolating cluster after the bond occupation.

At this moment, the calculation of S_i should be adjusted into

$$S_i = 1 - \prod_{j \in N_i} u_{i,j}, \tag{4}$$

and the percolating cluster fraction S would still be calculated by Eq. (3).

B. THE MAIN CAUSE OF THE INACCURACY OF THE ORIGINAL ALGORITHM

Let us consider a simple site percolation scenario where the message passing algorithm fails to produce accurate results. Think about an undirected triangle with nodes 1, 2, and 3 (see Fig.1). The incidence matrix for that triangle is as follows:

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

According to the incidence matrix and Eq. (1), we can list six self-consistent equations immediately. They are

$$\begin{aligned} u_{1,2} &= 1-p+p\cdot u_{2,3} \\ u_{2,1} &= 1-p+p\cdot u_{1,3} \\ u_{1,3} &= 1-p+p\cdot u_{3,2} \\ u_{3,1} &= 1-p+p\cdot u_{1,2} \\ u_{2,3} &= 1-p+p\cdot u_{3,1} \\ u_{3,2} &= 1-p+p\cdot u_{2,1} \end{aligned} \tag{5}$$

where p is the site occupation probability. Then, we should solves Eqs. (5) by numerical iterations, and substitute the solutions into the following equations

$$\begin{aligned} S_1 &= p-p\cdot u_{1,2}\cdot u_{1,3} \\ S_2 &= p-p\cdot u_{2,1}\cdot u_{2,3} \\ S_3 &= p-p\cdot u_{3,1}\cdot u_{3,2}. \end{aligned} \tag{6}$$

Finally, we can calculate the percolating cluster fraction S as follow:

$$S = \frac{S_1+S_2+S_3}{3}. \tag{7}$$

The leading eigenvalue of the non-backtracking matrix for that triangle is 1, and so we have $p_c = 1$. It means that for any $p(p < 1)$, Eqs. (5) only have a set of solutions $u_{i,j} = 1$. By substituting these solutions into Eqs. (6) and (7), we have $S = 0$. We can see that the result calculated by the message passing algorithm violates the fact that the triangle has a probability of p^3 to be completely preserved (see Fig. 1). As p value approaches 1, there is a greater discrepancy between the estimated value of S (zero) and its true value (p^3).

By scrutinizing Eqs. (1) and (5), we find that $u_{i,j}$ in every equation actually represents a mathematical expectation of many site occupation results. This design is very concise. However, it cannot guarantee compatibility with all possible outcomes, even when the probability of some outcomes is relatively high. For instance, considering that undirected triangle, a possible outcome denoted by the solutions $u_{i,j} = 0$ and $S = 1$ may not conform to Eqs. (5) for any $p(p < 1)$. We believe that this is the main cause of the inaccuracy of the message passing algorithm. Therefore, we will focus on analyzing several possible states after site or bond occupation, as long as these states have relatively high probabilities. Then we could calculate the expectation of the percolating cluster fractions in these states as the final estimate. This method may not be able to describe all possible states just with only one set of self-consistent equations like the original algorithm. However, its calculation procedure will fully consider those most possible states and reflects their impacts on the final result.

Fortunately, for every possible deterministic state, the message passing algorithm can achieve an accurate expectation. E.g., for that undirected triangle before the site occupation, we can list the following equations

$$\begin{aligned} u_{1,2} &= u_{2,3} \\ u_{2,1} &= u_{1,3} \\ u_{1,3} &= u_{3,2} \\ u_{3,1} &= u_{1,2} \\ u_{2,3} &= u_{3,1} \\ u_{3,2} &= u_{2,1}, \end{aligned} \tag{8}$$

and

$$\begin{aligned} S_1 &= 1-u_{1,2}\cdot u_{1,3} \\ S_2 &= 1-u_{2,1}\cdot u_{2,3} \\ S_3 &= 1-u_{3,1}\cdot u_{3,2}. \end{aligned} \tag{9}$$

Then, each deterministic state after the site occupation could be described by adjusting Eqs. (8). Here we would like to use $S_{(nj)}$ and $P_{(nj)}$ to denote the estimate of the total fraction of all clusters with loops when n_j nodes are occupied after the site occupation and the corresponding probability for such a state, respectively. The self-consistent equations for the state when all three nodes are occupied is the same as Eqs. (8), and by starting numerical iterations with $u_{i,j} = 0$ and substituting the solutions into Eqs. (9), we finally obtain $S(3) = 1$. For other states, we have $S(2) = 0$, $S(1) = 0$, and $S(0) = 0$. Considering their probabilities $P(3)$, $P(2)$, $P(1)$ and $P(0)$ (see Fig. 1 for details), we have $S = p^3$, which is equal to the true value. Obviously, real networks often have more complex structures than that triangle. However, this method can still be applied to such networks once its specific details are determined.

C. THE MODIFIED ALGORITHM FOR SITE PERCOLATION

Our modified algorithm for both site and bond percolation on undirected networks begins with listing self-consistent equations for each edge before the site or bond occupation. For edge (i -> j), we have

$$u_{i,j} = 1\cdot \prod_{k\in N_j\backslash i} u_{j,k}, \tag{10}$$

where $N_j\backslash i$ denotes the set of neighbors of j excluding i. If j has only one neighbor i, we will have $u_{i,j} = 1$. For this specific state, we have

$$S_i = 1-1\cdot \prod_{j\in N_i} u_{i,j} \tag{11}$$

and

$$S = \frac{\sum_{i=1}^{n} S_i}{n} \tag{12}$$

where N_i still denotes the set of direct neighbors of node i, and n is the total number of nodes.

Then, we will select l possible states to calculate the analytic value of the total proportion of clusters with loops after the site occupation at any specific site occupation probability p. The self-consistent equations for each state will be obtained by adjusting some parts of Eqs. (10), while keeping Eqs. (11)

and (12) unchanged. In particular, from here on we will no longer refer to the network typology.

Here we would like to use n_j to denote the total number of occupied nodes at the j-th possible state. Then we should decide which nodes will be occupied in each state. Obviously, even the value of n_j is fixed, its corresponding site occupation results will not be unique. The total number of possible site occupation results would be $\left(\binom{n}{n_j}\right)$, and actually we could only choose one of these results to represent all of them. Hence, the estimate of the percolating cluster fraction calculated by the chosen site occupation result should be close to the expectation of the estimates calculated by all those results. Some simple and straightforward approaches such as selecting the first or last n_j nodes from all those n nodes to occupy, may not always guarantee that. Here an ancient pseudo-random number generator RANDU proposed by IBM in 1961 [17], [18] is adopted due to its simplicity of operation, while many other more complex pseudo-random number generators may also work. The pseudo-random number generation mechanism of RANDU could be described by

$$X_{i+1} = (2^{16} + 3) \cdot X_i mod(2^{31}), \qquad (13)$$

where X_i ($1 \leq X_i \leq 2^{31}$-1) denotes a positive integer. If we give an integer in that range as the initial seed (e.g. $X_0 = n$), we will obtain a sequence of pseudo-random integers all in that range. Further, the range of those pseudo-random numbers can be adjusted arbitrarily, e.g. X_i could be converted to a positive integer $Y_i(1\leq Y_i \leq Z)$, where $\frac{2^{31}}{Z} \cdot Y_i > X_i$ and $\frac{2^{31}}{Z} \cdot (Y_i - 1) \leq X_i$. By this mechanism, we can sort all nodes in ascending order according to their identifiers, arbitrarily give the initial seed X_0 $\left(1 \leq X_0 \leq 2^{31} - 1\right)$ to produce X_1 and convert it to Y_1 $(1 \leq Y_1 \leq Z = n)$, and take the Y_1-th node out of the sequence as the first occupied one. Then we calculate X_2 by Eq. (13), and convert it to Y_2 $(1 \leq Y_2 \leq Z = n - 1)$ to obtain the second occupied node. That process will be repeated until we get n_j occupied nodes. According to the inherence of the pseudo-random generator, as long as X_0 remains unchanged, the subsequent pseudo-random numbers as well as the occupied nodes determined by them will also be fixed.

Given the nodes that are occupied in a specific state, we can make some adjustments to Eqs. (10) to obtain the self-consistent equations for that state. Let us focus on the left-hand of Eq. (10), if node i or j or both of them are unoccupied, it should be adjusted into $u_{i,j} = 1$. Otherwise, it should remain unchanged. By this simple adjustment mechanism, we will get self-consistent equations for that state. Solve them by iterations, and substitute the solutions to Eqs. (11) and (12), we will get the analytic value of the percolating cluster fraction for this state. Here we would like to represent that analytic value for a state where n_j nodes are occupied as $S(n_j)$. Take the previous undirected triangle as an example. For the possible state $n_j = 2$, and assuming that nodes 2 and 3 are occupied while node 1 is unoccupied by the pseudo-random generation, Eqs. (8) should be adjusted as follows. The preceding four sub-equations will be straightforward converted to $u_{1,2} = 1$, $u_{2,1} = 1$, $u_{1,3} = 1$ and $u_{3,1} = 1$. The last two sub-equations should be preserved. In this regard, we have $u_{2,3} = u_{3,1} = 1$ and $u_{3,2} = u_{2,1} = 1$. By substituting them to Eqs. (9) and (7), we get $S(2)$=0.

So far, for each of l selected states denoted by j $(1 \leq j \leq l)$, we can calculate $S(n_j)$, given the number of occupied nodes n_j in that state. Now we should determine $n_j(1 \leq j \leq l)$ to ensure that those l states have relatively high probabilities. Generally speaking, n_j should take an integer around $[n{\cdot}p]$, where [] means taking the integer part of the value. Let z $(z \geq 1)$ denotes the gap between n_{j-1} and n_j, and for an even l, we have

$$n_j = [n \cdot p] - (l/2 - j) \cdot z. \qquad (14)$$

z could be determined by the following formulas:

$$\sum_{i=[n\cdot p]+1-z\cdot l/2}^{[n\cdot p]+z\cdot l/2} \binom{n}{n_i} \cdot p^{n_i} \cdot (1-p)^{n-n_i} \geq 0.99$$
$$\sum_{i=[n\cdot p]+1-(z-1)\cdot l/2}^{[n\cdot p]+(z-1)\cdot l/2} \binom{n}{n_i} \cdot p^{n_i} \cdot (1-p)^{n-n_i} < 0.99. \qquad (15)$$

If we use the analytic value $S(n_j)$ of the state where n_j nodes are occupied to approximate that of the state where the occupied node number is between n_j-$z + 1$ and n_j, we could calculate the final estimate as follow:

$$S_{\text{final}} = \sum_{j=1}^{l} (S(n_j) \cdot \sum_{i=n_j-z+1}^{n_j} \binom{n}{n_i} \cdot p^{n_i} \cdot (1-p)^{n-n_i}). \qquad (16)$$

D. THE MODIFIED ALGORITHM FOR BOND PERCOLATION

Our modified algorithm for bond percolation is similar to site percolation in the following steps: First, we list self-consistent equations for the original network. These self-consistent equations can still be represented by Eq. (10). Then, we should determine the number of occupied edges for those l selected states after the bond occupation to guarantee that they have relatively high probabilities. Then, we still have to adjust Eq. (10) to obtain the new self-consistent equations for each of l selected states. We should also solve those self-consistent equations, and substitute the solutions into Eqs. (11) and (12) to calculate the analytic value of the total proportion of all clusters with loops for every state. Finally, we calculate the final estimate of the total proportion of all clusters with loops after the bond occupation as the expectation of the corresponding analytic values of those l selected states.

Here we would like to use m and m_j to denote the total number of edges and the number of occupied edges at the j- th possible state, respectively. Currently, p denotes the bond occupation probability, while other variables retain their previous definitions. The pseudo-random generator RANDU is also used to determine which edges are occupied at each possible state. This time we put all edges in a sequence, and take an edge out of the sequence every time according to the pseudo-random generation mechanism similar to the previous case of site percolation, until we get enough occupied edges.

The adjustment mechanism of Eq. (10) is as follow: we still have to first check the left-hand of that equation. If edge (i, j) is unoccupied, we should straightforward convert it to $u_{i,j} = 1$. Otherwise, it should be kept unchanged. Then by solving those self-consistent equations, and substituting the solutions into Eqs. (11) and (12), we could calculate $S(m_j)$. For example, let us think about the possible state $m_j = 2$ for bind occupation on the previous undirected triangle. Assuming that edge(1,2) (also edge (2,1)) is unoccupied while other edges are occupied by the pseudo-random generation, we should immediately convert the first two sub-equations of Eqs. (8) to $u_{1,2} = 1$ and $u_{2,1} = 1$. Then, $u_{1,3} = u_{3,2}$, $u_{3,1} = u_{1,2}$, $u_{2,3} = u_{3,1}$ and $u_{3,2} = u_{2,1}$ should be kept as they are. By the following iterations, we have $u_{2,3} = 1$, $u_{1,3} = 1$. Substitute those solutions into Eqs. (9) and (7), and we will get $S(2)$=0.

Parallel to the case of site percolation, we still select l possible states to calculate the analytic value of the total proportion of all clusters with loops, and set the occupied edge numbers of those states with an equal interval z. For an even l, we have

$$m_j = [m \cdot p] - (l/2 - j) \cdot z. \tag{17}$$

Now, z could be determined by the following formulas:

$$\sum_{i=[m\cdot p]+1-z\cdot l/2}^{[m\cdot p]+z\cdot l/2} \binom{m}{m_i} \cdot p^{m_i} \cdot (1-p)^{m-m_i} \geq 0.99$$
$$\sum_{i=[m\cdot p]+1-(z-1)\cdot l/2}^{[m\cdot p]+(z-1)\cdot l/2} \binom{m}{m_i} \cdot p^{m_i} \cdot (1-p)^{m-m_i} < 0.99. \tag{18}$$

Then, we could calculate the final estimate as follow:

$$S_{\text{final}} = \sum_{j=1}^{l} (S(m_j) \cdot \sum_{i=m_j-z+1}^{m_j} \binom{m}{m_i} \cdot p^{m_i} \cdot (1-p)^{m-m_i}). \tag{19}$$

III. APPLICATIONS TO REAL NETWORKS

Four concrete undirected networks are selected and downloaded from the well-known network data repository website Network Repository for comparison between the message passing algorithm and our modified version. The first network (available at: https://networkrepository.com/power-US-Grid.php) is the power grid of western states in the United States, the second one (available at: https://networkrepository.com/road-minnesota.php) provides the road network of Minnesota, the third one (available at: https://networkrepository.com/3elt.php) represents a 2D finite element network, while the last one (available at: https://networkrepository.com/08blocks.php) is a symmetric power graph of several graphs. Some common statistics of their characteristics are presented in Table 1. Here we would like to emphasize that if we use the total proportion of clusters with loops as the measure of the percolating cluster in numerical simulations, we will find

TABLE 1. Characteristics of four concrete networks.

networks	US power grid	Minnesota road network	2D finite element network	symmetric power graph
nodes	4940	2642	4720	300
edges	6594	3303	13722	576
density	0.0005	0.0009	0.0012	0.0128
maximum degree	19	5	9	36
average degree	2.6996	2.5004	5.8144	3.84
assortativity	0.0035	-0.1848	0.3107	-0.3641
average clustering coefficient	0.0801	0.0160	0.4111	0.6592

that the message passing algorithm provides very accurate estimates for most real networks. Those four networks are chosen mainly because the message passing algorithm may perform not so well on them.

When we apply our modified algorithm to each concrete network, we will first determine z value at $p = 0.5$ according to formulas (15) for site percolation and formulas (18) for bond percolation, respectively. That z value will remain unchanged when p take other values on the same network for the purpose of simplicity. For each specific p value, we will take the total number of nodes or edges as the initial seed ($X_0 = n$ for site percolation and $X_0 = m$ for bond percolation). The initial seed will be used to calculate the analytic value of the total proportion of all clusters with loops for each of l selected states one by one (the last X_i generated in the calculation of $S(n_j)$ or $S(m_j)$ will be used to generate the next pseudo-random integer in the calculation of $S(n_{j+1})$ or $S(m_{j+1})$), and based on them we can calculate S_{final} for that p value. The above specifications will ensure that the results remain consistent regardless of who operates our modified algorithm. The core pseudocode of our modified algorithm has been placed in the appendix section of this paper.

Fig. 2 presents results for site percolation on those four concrete networks. The first network US Power Grid has already been utilized by many researchers to examine their findings (e.g. [4], [19], [20], and in [4], the largest existing cluster was used as the indicator of the percolating cluster in numeral simulations). From Fig. 2(a), we can see that the estimates calculated by the message passing algorithm are always acceptable on that network. However, the discrepancies are relatively a bit large when the site occupation probability is

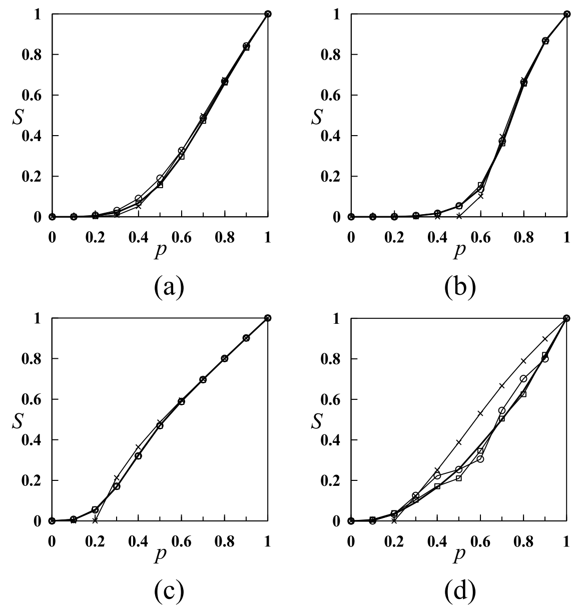


FIGURE 2. Site percolation results on four concrete networks. Solid-cross line denotes estimates of the message passing algorithm, solid-circle and solid-square lines denote estimates of the our modified algorithm when l=10 and 30, respectively, while bold solid line denotes the total fraction of nodes clusters with loops counted from numerical simulations and averaged over 1000 repetitions. The four networks are (a) Us Power Grid, (b) Minnesota Road Network, (c) 2D Finite Element Network and (d) Symmetric Power Graph.

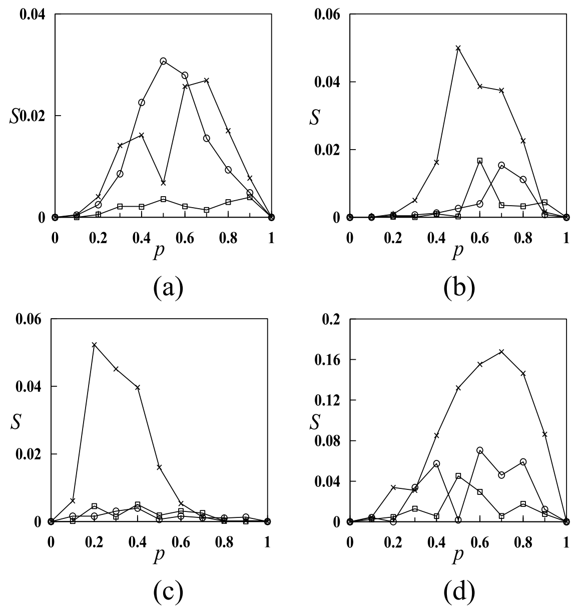


FIGURE 3. Absolute discrepancies of S estimates for site percolation on concrete networks. Solid-cross line denotes the absolute discrepancy of S estimates calculated by the message passing algorithm, while solid-circle and solid-square lines denote those calculated by our modified algorithm when l=10 and 30, respectively. The four networks denoted by (a), (b), (c) and (d) are the same as those in FIGURE 2.

between 0.6 and 0.8 (see Fig. 3(a) for details). In that range, some estimates calculated by our modified algorithm (when $l = 10$) are more accurate than those calculated by the original algorithm when others are not. Totally speaking, the accuracy of our modified algorithm (when $l = 10$) is close to the original algorithm in that range. However, if the number of the selected states l is raised to 30, the accuracy of our modified algorithm will be significantly improved, and its estimates seem to be much closer to the simulation results than those of the original algorithm.

From Fig. 2(b) and (c) as well as Fig. 3(b) and (c), we find that the original algorithm performs not well when p takes some values (from 0.5 to 0.7 on the second network, and from 0.2 to 0.4 on the third network), and at those p values, the estimates of our modified algorithm (both when $l = 10$ and $l = 30$) are always accurate. The last network is somewhat special as it is mainly composed by many triangles tied together, and the original algorithm estimates are not accurate at most p values (from 0.4 to 0.9) on it (see Fig. 2(d) and Fig. 3(d)). Further by our observation, the original message passing algorithm is bound to face challenges when applied to networks primarily composed of short loops, as exemplified by the previously mentioned undirected triangle and the final real network. In contrast, our modified algorithm significantly outperforms the original algorithm in both cases. Based on all the analysis above, we find that the estimates of our modified algorithm (when $l = 30$) are always close to the simulation results. However, we still cannot conclude that the estimate of our modified algorithm (when $l = 30$) is more accurate than that (when $l = 10$) at any p value. For instance, the estimate (when $l = 10$) is closer to the simulation value than that (when $l = 30$) at $p = 0.5$ on the last network.

The results for bond percolation on those real networks are presented in Fig. 4, while their corresponding absolute discrepancies are presented in Fig. 5. We can draw conclusions similar to the case of site percolation such as at most p values, our modified algorithm (both when $l = 10$ and $l = 30$) perform better than the origin one. Or more exactly, considering both the cases of site and bond percolation, we could conclude that at some p values where the original message passing algorithm performs good, our modified algorithm also works well, and at this moment we cannot say the estimates of which algorithm are always more accurate. However, at those p values where the original message passing algorithm performs not such good (e.g. the absolute discrepancy of the estimate calculated by the original algorithm is higher than around 0.05), the estimates of our modified algorithm are much more accurate. Especially, the estimates of our modified algorithm (when $l = 30$) are always close to the simulation results at all p values on all networks.

IV. DISCUSSIONS

Our modified algorithm is semi-analytic as a pseudo-random number generation is adopted in estimation of the percolating cluster. Except for the first step listing self-consistent equations before the site or bond occupation, we do not need to refer to the specific network topology in the subsequent procedures. This is the similarity between the original message

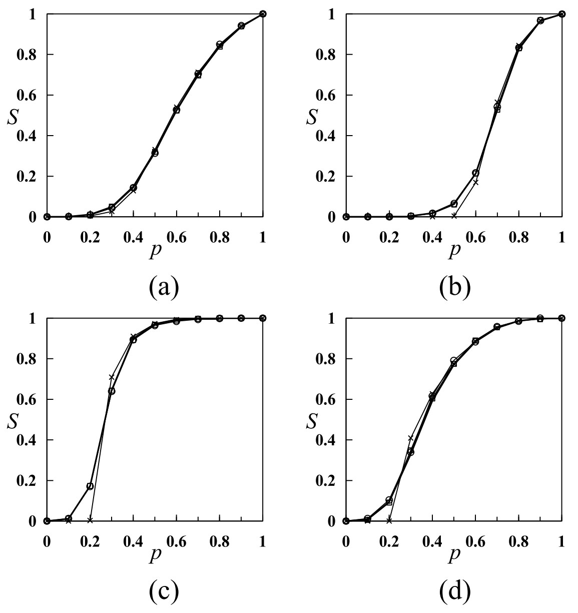


FIGURE 4. Bond percolation results on four concrete networks. Those networks and symbols are the same as their counterparts in FIGURE 2.

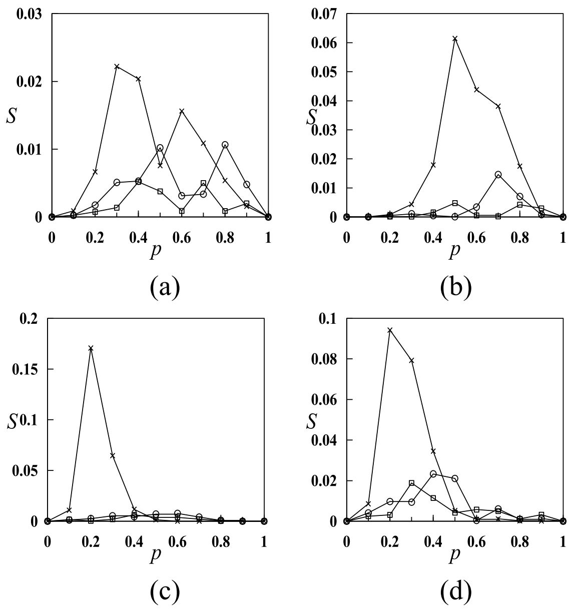


FIGURE 5. Absolute discrepancies of S estimates for bond percolation on concrete networks. Those networks and symbols are the same as their counterparts in FIGURE 3.

passing algorithm and our modified version. Further, as long as the initial pseudo-random seed X_0 keep unchanged, the estimates calculated by our modified algorithm will remain the same. Even we arbitrarily adopt different initial seeds, the overall accuracy of those estimates seems not to fluctuate much.

Parameter l determines the number of states selected by our modified algorithm to calculate S_{final}. Since for each state, we have to obtain a set of new self-consistent equations by adjusting the initial equations and then solve them, the overall calculation amount of our modified algorithm will be in proportion to l. Considering that the original message passing algorithm only has to list and solve one set of such equations, the calculation amount of our modified algorithm is around ten times that of the original one when $l = 10$, and it increases to thirty times when $l = 30$. Numerical simulations indicate that, with $l = 10$, our modified algorithm ensures better overall accuracy than the original one on most networks. Additionally, with $l = 30$, the estimates of our modified algorithm consistently closely match the simulation values.

V. CONCLUSION

In summary, we developed a semi-analytic algorithm to estimate the proportion of the percolating cluster by modifying the message passing algorithm proposed by Karrer et al. and introducing in a simple pseudo-random number generation mechanism. Like the original algorithm and other percolation analytic models, our modified algorithm can be applied to various real networks to estimate the fraction of the percolating cluster when their nodes or edges experience random attacks or failures. For instance, this is relevant in situations where a considerable number of servers in the Internet fail due to DDoS attacks or numerous roads in a traffic network become impassable during the morning peak due to congestion.

By comparing the estimates obtained from the original and modified algorithms with the simulation results, we found that our modified algorithm consistently produces estimates more accurate than those of the original algorithm. This is achieved under the condition of selecting a sufficient number of states to calculate the final result. The calculation mechanism of our modified algorithm determines that its estimates are always higher than zero even when the occupation probability is extremely small. Those estimates may change with the initial pseudo-random seed, which lead to that this algorithm may not provide the exact value of the percolation threshold like other percolation analytic models or algorithms. Nevertheless, our modified algorithm was designed to address the limitations of the original message passing algorithm, particularly its performance issues on networks with many short loops. Its aim is to offer a more accurate estimation of the percolating cluster on any real network. We have primarily achieved these goals with our modified algorithm.

APPENDIX

Here we present the core pseudocode of our modified algorithm. The corresponding programs, used to list self-consistent equations for the selected state under both site percolation and bond percolation, have been uploaded to the website Zenodo. Please refer to https://zenodo.org/records/10672999.

The function '**(int) prandom: (int)** ***ub***' is primarily based on the pseudo-random number generator RANDU. This function takes a positive integer ub as input and returns a random number between 1 and ub. It can be applied to scenarios such as randomly selecting one node from a set of n nodes to occupy. Its design is primarily based on the Eq. (13) and its related discussions in the main text. The core pseudocode is outlined below:

```
X0 <- ($2^{16}$+3)*X0%$2^{31}$ //the initial value of X0, which
                             is also the random seed, will be
                             provided by the user
for i <- 1 to ub
{
if (double)$2^{31}$/ub*i > X0
break
}
return i
```

The following program will list the self-consistent equations for each selected state after the node occupation. The main parameters, p (site occupation probability), z (the gap in the number of occupied nodes between two selected states), X0 (random seed), j (the selected state identifier), and l (the total number of selected states), should be provided by the user. The meanings of these parameters are consistent with those explained in the main text. Additionally, the program requires inputting the bond data file of the target network, where the number of nodes n and edges m are determined by the target network. The core pseudocode is as follows:

```
$n_j$ <- n*p-(l/2-j)*z //$n_j$ denotes the number of nodes
                        occupied in the current state, and
                        its calculation is primarily based
                        on Eq.(14)
for i <- 1 to n
{
px1[i] <- i //place all nodes into the array $px1$
             according to their identifiers
px2[i] <- 0 //px2[i] represents the occupation status
             of node $i$, where 0 indicates
             unoccupied, and 1 indicates occupied.
}
fr <- n //fr represents the number of elements
         in $px1$, with an initial value of $n$
for <- 1 to $n_j$ //selecting one node to occupy each time
{
k <- [prandom: fr]
px2[px1[k]] <- 1 //update the status of the $k$-th
                  element in $px1$ to occupied
for jj <- k to fr-1 //remove the $k$-th element from $px1$
px1[jj]=px1[jj+1]
fr <- fr-1
}
cpt1 <-fopen("output.txt","w");//write the self-
                                consistent equations
                                one by one into
                                the file output.txt
for i <- 1 to n
for j <- 1 to n //here j represents a loop variable
if am[i][j]==1&&i!=j //if there is an edge between
                      nodes $i$ and $j$, the value of
                      $am[i][j]$ is 1; otherwise, it is 0.
{
 if px2[i]==1&&px2[j]==1
 {
 fprintf(cpt1,"u%d_%d=1*",i,j);
 for k <- 1 to n
 if am[j][k]==1&&k!=i
 fprintf(cpt1,"u%d_%d*",j,k);
 fprintf(cpt1,"1,");
  }
 else
fprintf(cpt1,"u%d_%d=1,",i,j);
}
```

REFERENCES

[1] P. Mann, V. A. Smith, J. B. O. Mitchell, C. Jefferson, and S. Dobson, "Exact formula for bond percolation on cliques," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 104, no. 2, Aug. 2021, Art. no. 024304, doi: 10.1103/physreve.104.024304.

[2] M. Li, R.-R. Liu, L. Lü, M.-B. Hu, S. Xu, and Y.-C. Zhang, "Percolation on complex networks: Theory and application," *Phys. Rep.*, vol. 907, pp. 1–68, Apr. 2021, doi: 10.1016/j.physrep.2020.12.003.

[3] T. Fu, Y. Zhang, and C. Li, "Two typical analytic models for reverse bond percolation on real networks," *Phys. A, Stat. Mech. Appl.*, vol. 625, Sep. 2023, Art. no. 129029, doi: 10.1016/j.physa.2023.129029.

[4] T. Fu, L. Zou, C. Li, and J. Zhao, "A relatively simple model for percolation properties of real networks," *Phys. Lett. A*, vol. 381, no. 32, pp. 2578–2582, Aug. 2017, doi: 10.1016/j.physleta.2017.06.005.

[5] B. Karrer and M. E. J. Newman, "Message passing approach for general epidemic models," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 82, no. 1, Jul. 2010, Art. no. 016101, doi: 10.1103/physreve.82.016101.

[6] B. Karrer, M. E. J. Newman, and L. Zdeborová, "Percolation on sparse networks," *Phys. Rev. Lett.*, vol. 113, no. 20, Nov. 2014, Art. no. 208702, doi: 10.1103/physrevlett.113.208702.

[7] F. Radicchi, "Predicting percolation thresholds in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 91, no. 1, Jan. 2015, Art. no. 010801, doi: 10.1103/physreve.91.010801.

[8] R. Kühn, "Disentangling giant component and finite cluster contributions in sparse random matrix spectra," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 93, no. 4, Apr. 2016, Art. no. 042110, doi: 10.1103/physreve.93.042110.

[9] G. Bianconi and F. Radicchi, "Percolation in real multiplex networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 94, no. 6, Dec. 2016, Art. no. 060301, doi: 10.1103/physreve.94.060301.

[10] G. Bianconi, "Epidemic spreading and bond percolation on multilayer networks," *J. Stat. Mechanics: Theory Exp.*, vol. 2017, no. 3, Mar. 2017, Art. no. 034001, doi: 10.1088/1742-5468/aa5fd8.

[11] H. Peng, C. Qian, D. Zhao, M. Zhong, J. Han, T. Zhou, and W. Wang, "Message-passing approach to higher-order percolation," *Phys. A, Stat. Mech. Appl.*, vol. 634, Jan. 2024, Art. no. 129446, doi: 10.1016/j.physa.2023.129446.

[12] R. Cohen, K. Erez, D. ben-Avraham, and S. Havlin, "Resilience of the internet to random breakdowns," *Phys. Rev. Lett.*, vol. 85, no. 21, pp. 4626–4628, Nov. 2000, doi: 10.1103/physrevlett.85.4626.

[13] F. Radicchi and C. Castellano, "Beyond the locally treelike approximation for percolation on real networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 93, no. 3, Mar. 2016, Art. no. 030302, doi: 10.1103/physreve.93.030302.

[14] K. I. Hashimoto, "Zeta functions of finite graphs and representations of p-adic groups," in *Automorphic Forms Geometry Arithmetic Varieties*, vol. 15, K. Hashimoto and Y. Namikawa, Eds. New York, NY, USA: Academic, 1989, pp. 211–280, doi: 10.1016/B978-0-12-330580-0.50015-X.

[15] F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborová, and P. Zhang, "Spectral redemption in clustering sparse networks," *Proc. Nat. Acad. Sci. USA*, vol. 110, no. 52, pp. 20935–20940, Dec. 2013, doi: 10.1073/pnas.1312486110.

[16] F. Radicchi, "Percolation in real interdependent networks," *Nature Phys.*, vol. 11, no. 7, pp. 597–602, Jul. 2015, doi: 10.1038/nphys3374.

[17] J. W. Bang, R. E. Schumacker, and P. L. Schlieve, "Random-number generator validity in simulation studies: An investigation of normality," *Educ. Psychol. Meas.*, vol. 58, no. 3, pp. 430–450, Jun. 1998, doi: 10.1177/0013164498058003005.

[18] E. A. Luengo, "A brief and understandable guide to pseudo-random number generators and specific models for security," *Statist. Surv.*, vol. 16, pp. 137–181, Jan. 2022, doi: 10.1214/22-ss136.

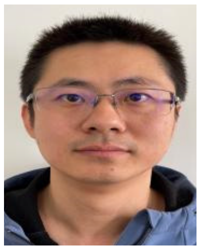
[19] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998, doi: 10.1038/30918.

[20] T. Fu, R. Sun, C. Li, and L. Wu, "Node differentiation protection concerning model of localized attack on real networks," *Phys. A, Stat. Mech. Appl.*, vol. 526, Jul. 2019, Art. no. 120947, doi: 10.1016/j.physa.2019.04.183.

CHENGUANG LI received the Ph.D. degree in management science and engineering from Beijing University of Technology, in 2014. He is currently an Associate Professor and a Master Tutor with the Economics and Management School, North China University of Technology. His research interests include economic policy, complex networks, and multi-agent simulation.

TAO FU received the Ph.D. degree in management science and engineering from Beijing University of Technology, in 2011. He is currently an Associate Professor and a Master Tutor with the Economics and Management School, Beijing University of Technology. His research interests include complex networks, social networks, and multi-agent simulation.

LIAN WANG received the Ph.D. degree in management science and engineering from the Central University of Finance and Economics, in 2010. She is currently a Lecturer with the Economics and Management School, Beijing University of Technology. Her current research interests include information system analysis and design, financial management, and software engineering.



RAN SUN received the master's degree in software engineering from Beijing University of Technology, in 2015. He is currently a Senior Engineer with the Computer School, North China Institute of Aerospace Engineering. His current research interests include software engineering, complex networks, and computer simulation.