Hindawi
Complexity
Volume 2021, Article ID 6672630, 17 pages
https://doi.org/10.1155/2021/6672630

Research Article

Predicting Spread Probability of Learning-Effect Computer Virus

Wei-Chang Yeh,[1] **Edward Lin**,[2] **and Chia-Ling Huang**[3]

[1]*Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 300, Taiwan*
[2]*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1234, USA*
[3]*Department of International Logistics and Transportation Management, Kainan University, Taoyuan 33857, China*

Correspondence should be addressed to Wei-Chang Yeh; wcyeh@ie.nthu.edu.tw

Received 23 November 2020; Revised 4 February 2021; Accepted 7 June 2021; Published 12 July 2021

Academic Editor: Luxing Yang

With the rapid development of network technology, computer viruses have developed at a fast pace. The threat of computer viruses persists because of the constant demand for computers and networks. When a computer virus infects a facility, the virus seeks to invade other facilities in the network by exploiting the convenience of the network protocol and the high connectivity of the network. Hence, there is an increasing need for accurate calculation of the probability of computer-virus-infected areas for developing corresponding strategies, for example, based on the possible virus-infected areas, to interrupt the relevant connections between the uninfected and infected computers in time. The spread of the computer virus forms a scale-free network whose node degree follows the power rule. A novel algorithm based on the binary-addition tree algorithm (BAT) is proposed to effectively predict the spread of computer viruses. The proposed BAT utilizes the probability derived from PageRank from the scale-free network together with the consideration of state vectors with both the temporal and learning effects. The performance of the proposed algorithm was verified via numerous experiments.

1. Introduction

Almost all individuals and industries constantly rely on computer and network technology [1–16]. Thus, the types of computer viruses are becoming more diversified, and the intensity of the attacks is increasing. When a computer virus infects a facility, the virus seeks to invade other facilities in the network by exploiting the convenience of the network protocol and the high connectivity of the network. Hence, computer viruses cause serious damage to computer facilities. The terminology "computer virus" was introduced by von Neumann in 1966 [17], and it was extended from his lectures regarding self-reproducing computer programs given in 1949 [18]. Subsequently, scholars performed small-scale research on computer viruses, publishing conference papers from 1988 to 2000, and numerous computer virus-related research papers have been published in different journals. For example, Risak studied the functional virus in computer programs in 1972 [19], and Kraus researched self-reproducing computer programs and investigated the behavior of computer-language-like biological viruses in 1980 [20].

The evolution of terminologies on virus includes the worm, malware (trojan horse is the old term) that originated approximately in 1990, and advanced persistent threats (APT) that are newer developments on computer viruses in the past decade.

Researches on worm involve Shoutkov and Spesivtsev explored self-replication of worm [21], and Griffin and Brooks paid attention to the spread of worm in the computer network [22]. The computer viruses of malware are researched such as Witte explored the detection of malware [23] and Peng et al. focus on a survey of the propagation of malware [24]. Furthermore, the researches in APT including Li and Yang, improved the system of cloud memory under APT [25] and Tian et al. defended against APT in power grid [26].

In the past five years, research on computer viruses has focused on defense against viruses [27–29]. The research on computer viruses can be classified into the following major

research directions: the description of the computer virus, the detection of computer viruses, and protection against computer viruses:

1.1. The Description of the Computer Virus. Eichin and Rochlis described the computer virus via a detailed analysis in 1989 [30], and Spafford investigated how computer viruses are formed in 1994 [31].

1.2. The Detection of Computer Viruses. Davis investigated the detection of computer viruses to enhance the control of risk management in 1988 [32]. Okamoto and Masumoto adopted authentication for detecting computer viruses in 1990 [33]. Spinellis proved that the detection of computer viruses is NP-complete [34]. A detection mechanism for processing sign streams was adopted to evaluate viruses by Wang et al. in 2015 [35]. A model for the nonlinear vaccination probability was used to detect computer viruses by Gan et al. in 2004 [36].

1.3. Protection against Computer Viruses. Al-Dossary proposed a classification formula for defense against computer viruses in 1989 [37]. Yuan et al. established a virus model to optimize the performance of the infection mechanism in 2009 [38]. Youssef and Scoglio focused on reducing the dissemination of viruses by optimizing the weight function of the network structure in 2014 [39].

Understanding the development of computer viruses is important for understanding the historical defense strategies against viruses [27]. In addition, for the detection of the aforementioned computer viruses, the method of identifying the code mode of the virus can be used. If the virus has a self-replicating function whereby the code of the virus is copied to other files, the appropriate protection strategy for the self-replicating virus can be selected and executed immediately [34]. Therefore, the aforementioned three types of computer-virus research all involve protection against computer viruses.

The lifecycle of the virus includes dormant, propagation, triggering, and execution. The dormant phase indicates that the computer virus code is being created and finally born. The propagation phase shows that the files of computer viruses are placed in places that are easy to propagate. Once infected by the computer viruses, it will cause great harm. In addition, the triggering and execution indicate when all the conditions are formed; the computer viruses then begin to execute destructive actions.

Therefore, to prevent computer virus infection, protection in advance is important in the propagation phase. Computer viruses persist owing to the constant demand for computer networks. Hence, it is important to predict the probabilities of the virus spreading to different areas so as to interrupt the relevant connections between the uninfected and possibly infected computers in time to save the files in the computers. This was the major focus of this study.

With the strength of the advanced technology, the susceptible-infectious-recovered (SIR) model is adapted to predict the number of computer-virus spread areas in this study. In the SIR model, all susceptible nodes can be infected at most once. After the infected node is recovered by removing the computer virus, it is protected by using a virus detection and killer software such that it is impossible to be infected by the same computer virus.

Assume that a virus can propagate in a heterogeneous environment freely. Hosts like Linux, Windows, Mac, etc., and different operating systems can have different code bases and the same virus may not work for all OS.

The spread of computer viruses is a scale-free model formed by a scale-free network in which all node degrees follow a power-law distribution. The PageRank algorithm is the most popular among the different scale-free model-related algorithms for calculating the influence of nodes. Hence, the PageRank algorithm is used to provide the theoretical spread probability of the computer virus.

With the defense mechanism (antivirus protection) in the scale-free model to prevent or slow down virus propagation, the computer virus can be detected and killed more easily from time to time. The above defense mechanism is called the learning effect here. Hence, the learning effect can be the temporal learning-effect spread probability $p_{i,j,t}$ of the computer virus spread from nodes $i \in V$ to $j \in V(i)$ and is higher than $p_{i,j,\tau}$ if $\tau < t$ during the computer-virus propagation.

During the process of computer-virus propagation, consecutive timeslots in the time period when the infected node i can still spread out the computer virus are called valid timeslots. An infected node can affect any neighboring node at any valid timeslot; that is,

$$\sum_{t} \sum_{\forall i \in V} \sum_{k \in V(i)} p_{i,k,t} = 1 \text{ for any valid timeslot } t, \tag{1}$$

$$0 \le p_{i,k,t} \le 1, \tag{2}$$

for any node k in $V(i)$ and the validation timeslot t.

A novel computer-virus spread dynamic model based on the binary-addition tree (BAT) search algorithm with the learning effect is proposed for modeling the spread of computer viruses. The BAT proposed by Yeh [40] is a heuristic search method similar to the depth-first search (DFS), breadth-first search (BFS), and universal generating function methodology (UGFM). The BAT is more efficient than the DFS and more economical with regard to computer memory than the BFS and UGFM, both of which can crash the computer system because of computer memory overflow problems. Moreover, the BAT is easy to learn, convenient to code, and flexible (i.e., it can be made-to-fit).

The objective of this study was to theoretically predict the probabilities of a computer virus infection and the spread of the virus to different areas. The remainder of the paper is organized as follows. Section 2 provides acronyms and notations. Section 3 presents an overview of the infection model, the scale-free model, the PageRank algorithm, the BAT, and the learning effect, which form the basis of the proposed dynamic BAT. Section 4 introduces the novel temporal learning-effect spread probability and period, which are required data for using the proposed dynamic

BAT. Section 5 describes the proposed state vectors formed by the spread vector and the temporal that needed to be found in the proposed dynamic BAT before predicting the spread areas of computer viruses. Section 6 formally presents the proposed dynamic BAT, together with its computational complexity, a demonstration, and experimental results. Section 7 concludes the paper.

2. Acronyms and Notations

All required acronyms and notations are provided in Tables 1 and 2, respectively.

Table 1: Acronyms.

APT	Advanced persistent threats
SIR	Susceptible-Infectious-Recovered Model
BAT	Binary-Addition Tree
BFS	Breadth-First Search
DFS	Depth-First Search
UGFM	Universal Generating Function Methodology

3. Infection Model, Scale-Free, Page Rank, BAT, and Learning Effect

The proposed dynamic BAT, which is a scale-free model, is based on the BAT with a temporal learning effect to predict the areas infected by the computer virus and the areas to which the virus will spread. The infection model is adopted to describe the spread of computer viruses. The scale-free model and PageRank algorithm are used to simulate the computer-virus spread probability before the proposed dynamic BAT is used to predict the probability of the areas infected and spread from the computer virus and the learning effect, which is integrated into the proposed model to simulate the spread of the computer virus in a more practical manner.

Hence, before the proposed BAT is discussed, an overview of the infection model, the scale-free model, the PageRank algorithm, the traditional BAT, and the learning effect are described in this section.

3.1. Infection Model. In recent years, the theory of the spread of epidemics in complex networks has yielded considerable success. Individuals in the system have several basic states: the susceptible state S (healthy but may be infected); infected state I; and removal state R (infected after being cured and gaining immunity or dying after infection).

There are three mature epidemic infection models: the Susceptible–Infected (SI) model, Susceptible–Infected–Susceptible model, Susceptible–Infectious–Recovered (SIR) model, and Susceptible–Infectious–Recovered–Susceptible model.

In the SIR model, a susceptible node (S) has become infectious (I) and can recover (R) to obtain lifelong immunity after curing. The SIR model is the most popular mathematical model. The spread of computer viruses is similar to that of epidemics, and both propagations can be captured by a scale-free model. Hence, the SIR model was adopted to model the spread of the computer virus.

The SIR model acquires lifelong immunity after an illness as shown in Figure 1 [41]. Let $S(t)$, $I(t)$, and $R(t)$ be the proportions of susceptible, infectious, and recovered nodes, respectively, and $S(t) + I(t) + R(t) = 1$ in $G(V, E)$ [41, 42]. The differential equation describing the propagation mechanism of the SIR model is as follows [41, 42]:

$$\begin{aligned} \frac{\mathrm{d}S(t)}{\mathrm{d}t} &= -\beta \cdot S(t) \cdot I(t), \\ \frac{\mathrm{d}I(t)}{\mathrm{d}t} &= -\beta \cdot S(t) \cdot I(t) - \gamma(t), \\ \frac{\mathrm{d}R(t)}{\mathrm{d}t} &= \gamma(t), \end{aligned} \tag{3}$$

where β and γ represent the transmission and recovery rates, respectively.

3.2. Scale-Free and Page Rank Algorithm. The scale-free network is a special network, and its growth is independent of the number of nodes with the same underlying structure. The major difference between the scale-free network and other networks is that it has power-law (or scale-free) degree distributions. For example, the network in Figure 2 is generated from the Barabási–Albert model, which is the first scale-free model.

PageRank, which is used by Google search engines, was the first and the most popular among these famous algorithms for ranking nodes (web page, website, user, etc.) according to importance in the scale-free network. The PageRank value of a node (web page, website, user, etc.) $i \in V$ is the probability that users clicking on nodes randomly will arrive at i. The array PR can be calculated as follows:

$$\mathrm{PR} = \left[d \cdot M + \frac{(1-d)}{N_{\mathrm{node}}} \cdot I \right] \cdot \mathrm{PR}, \tag{4}$$

where PR (i) represents the i^{th} element in PR for all $i \in V$, N_{node} represents the number of nodes, d represents a damping factor between 0 and 1, M represents the normalized adjacency square matrix such that $\sum_{a=1}^{N_{\mathrm{node}}} M_{a,b} = 1$, $M_{a,b}$ represents the element in the a^{th} row and the b^{th} column in M, and I represents the identity matrix. The sizes of both matrices I and M are $\mathrm{N}_{\mathrm{node}} \times \mathrm{N}_{\mathrm{node}}$.

The pseudo code of the PageRank algorithm is described as follows (Algorithm 1).

STEP PR0 initializes PR (i) for all $i \in V$. STEP PR1 updates PR (i) for all $i \in V$ according to Equation (4), by adding part of PR (j) for all nodes $j \in V$ with $e_{j,i} \in V$. STEP PR2 adjusts the value of PR (i) by assuming that the search of users will continue even if it reaches a dead end for all $i \in V$. When PR (i) is updated and redistributed in STEPs PR1 and PR2 recursively, the value of PR (i) converges, and the process halts for all $i \in V$.

For example, in Figure 2, we need to have the adjacency matrix first and normalize the matrix by dividing the

Table 2: Notations.

$G(V, E)$	A scale-free network with sets of nodes V and arcs E
N_{node}	The number of nodes
$e_{i,\ j}\ e_{i,j} \in E$	Such that information can be transmitted directly from node i to j
$Deg(i)$	Degree of node i
$V(i)$	Subset of nodes that receive information from node i
t_i	Infected timeslot of node $i \in V$
$p_{i,j}$	Spread probability that the computer virus is spread out from an infected node $i \in V$ to a susceptible node $j \in V(i)$
$p_{i,j,t}$	Temporal learning-effect spread probability of the computer virus from nodes $i \in V$ to $j \in V(i)$ for any valid timeslot t
$S(t)$	Proportions of susceptible nodes
$I(t)$	Proportions of infectious nodes
$R(t)$	Proportions of recovered nodes
β	Transmission rates
γ	Recovery rates
PR	Probability that users clicking on nodes randomly will arrive at i
PR (i)	The i^{th} element in PR for all $i \in V$
d	A damping factor between 0 and 1
M	Normalized adjacency square matrix
$M_{a,b}$	Element in the a^{th} row and the b^{th} column in M
I	Identity matrix
X	State vector
$X(i)$	Value in the i^{th} coordinate of vector X
T	Timeslot vector
$T_{i,t}$	The t-lag temporal vector of node i
Y	Temporal vector
TARGET	The first infected node
N_t	The number of temporal state vector candidates for time lag t
n_t	The number of feasible temporal vectors for time lag t

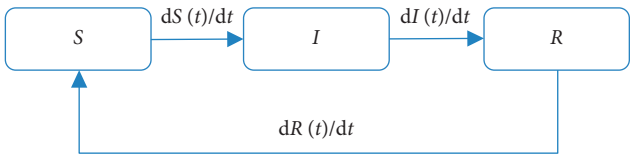

Figure 1: Example network.

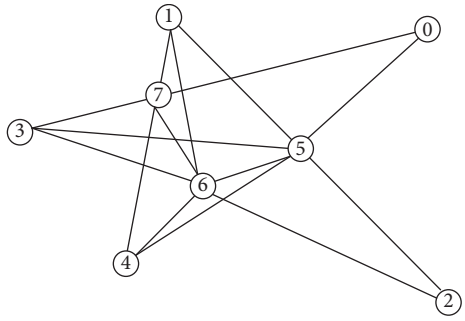

Figure 2: Example network.

element values in the adjacency matrix by their degrees, as shown in Tables 3 and 4, respectively.

Then, by simply following the pseudocode above, we obtain all the PR (i) values for all $i \in V$, as follows:

As shown in Table 5, a higher degree of the node corresponds to a higher probability to have a larger PR value. Hence, more important nodes have more links from other nodes in the PageRank algorithm, which is the "rich-gets-richer" phenomenon [41].

PR (i) is used to simulate the spread probability of the computer virus to node i in the proposed dynamic BAT to solve the spread of the computer-virus problem.

3.3. BAT. The BAT proposed by Yeh is a simple implicit enumeration method. Experiments revealed that the BAT is more efficient than the DFS and more economical with regard to computer memory than the BFS and UGFM. The DFS, BFS, and UGFMs are all well-known implicit enumeration methods.

By adding one to the zero vector repeatedly via binary addition, the BAT can generate all binary-state vectors whose coordinates are either 0 or 1. Let X be a binary-state vector with an n-tuple and X_i be the value of its i^{th} coordinate (Algorithm 2). The source code of the BAT is presented below [40]:

In STEP B0, the BAT begins to generate all vectors from the zero vector X. From STEPs B1 to B3, the state vector X is added to generate a new vector repeatedly. To reduce the runtime, the current coordinate is changed either from 0 to 1 or from 1 to 0. If it is changed from 1 to 0, we must go to the coordinate adjacent to the current coordinate to repeat the same procedure until it is changed from 0 to 1. After each new X is generated in STEP B4, its probability, cost, time, or any predefined function can be calculated. STEP B4 also tests whether the stopping criterion SUM $= n$ is satisfied, that is, whether X becomes a vector of which all the coordinates are 1.

For example, let $n = 5$ and $X = (0, 0, 0, 0, 0)$. To easily understand how BAT is based on the binary addition, each vector is rewritten to a binary number such that the i^{th} digit

(i) INPUT: A scale-free network $G(V, E)$.
(ii) OUTPUT: The PageRank value PR (i) for all $i \in V$.
(iii) STEP PR0. Let $t = 0$ and PR $(i) = 1/n$ for all $i \in V$.
(iv) STEP PR1. Let PR $= [d \cdot M + (1-d)/\mathrm{N_{node}} \cdot I] \cdot$PR.
(v) STEP PR2. Let PR (j) = PR(j) + $\sum_{i \in I \subseteq V}PR(i)/|V-I|$, where $I = \{i \in V \mid \mathrm{Deg_{out}}(i) = 0\}$ and for all $j \in (V-I)$.
(vi) STEP PR3. Halt if there is no change of PR (i) for all $i \in V$. Otherwise, let $t = t + 1$ and go to STEP PR1.

Algorithm 1: PageRank algorithm.

Table 3: The adjacency matrix.

i	0	1	2	3	4	5	6	7
0						1		1
1						1	1	1
2						1	1	
3						1	1	1
4						1	1	1
5	1	1	1	1	1		1	
6		1	1	1	1	1		1
7	1	1		1	1		1	
Deg (i)	2	3	2	3	3	6	6	5

Table 4: The normalized adjacency matrix based on each column.

i	0	1	2	3	4	5	6	7
0	0	0	0	0	0	1/6	0	1/5
1	0	0	0	0	0	1/6	1/6	1/5
2	0	0	0	0	0	1/6	1/6	0
3	0	0	0	0	0	1/6	1/6	1/5
4	0	0	0	0	0	1/6	1/6	1/5
5	1/2	1/3	1/2	1/3	1/3	0	1/6	0
6	0	1/3	1/2	1/3	1/3	1/6	0	1/5
7	1/2	1/3	0	1/3	1/3	0	1/6	0

of such a binary number is equal to the i^{th} coordinate of X, e.g., the binary number of $X = (0, 0, 0, 0, 0)$ is 00000. Note that there 32 different vectors in total because $2^5 = 32$.

Following the BAT code listed above, we have the first five new state vectors generated from zero in the sequence:

$$\begin{aligned} &00000 + 1 = 00001,\\ &00001 + 1 = 00010,\\ &00001 + 1 = 00011,\\ &00011 + 1 = 00100,\\ &00100 + 1 = 00101. \end{aligned} \tag{5}$$

In the same way, we have all state vectors from (0, 0, 0, 0, 0) to (1, 1, 1, 1, 1) obtained from the BAT without duplications as listed in Table 6.

From the above, the BAT is very simple to learn, easy to code, and flexible (can be made-to-fit). Hence, BAT is modified to solve the proposed problem.

3.4. Learning Effect. In economics, productivity increases and results in higher wages after suitable education, and this process is the learning effect. In realistic industrial processes, the operation time is reduced because the workers' skill or the flow process improves steadily, and this phenomenon is also called the learning effect. In many real-world applications [7, 28, 29, 40, 43–49], the learning effect is pragmatic. Hence, the learning effect is introduced in this work to study the proposed computer-virus spread area prediction problem and offers a defense mechanism (antivirus protection) in the scale-free model to prevent or slow down virus propagation.

Table 5: The Deg (i), $V(i)$, $C(i)$, and PR (i) values of node i.

i	Deg (i)	$V(i)$	PR (i)
0	2	{5, 7}	0.117460
1	3	{5, 6, 7}	0.119714
2	2	{6, 7}	0.117062
3	3	{5, 6, 7}	0.119714
4	3	{5, 6, 7}	0.119714
5	6	{0, 1, 2, 3, 4, 6}	0.138452
6	6	{1, 2, 3, 4, 5, 7}	0.135284
7	5	{0, 1, 3, 4, 6}	0.132599

Let $p_{i,j,t}$ be the probability that the computer virus spreads from node i to its j^{th} state at time t. If there is no learning effect, $p_{i,j,t}$ is a constant for all values of t. However, $p_{i,j,t}$ is reduced occasionally because users acknowledge the spread of the computer virus and learn how to prevent infection or propagation after the infection.

The values of $p_{i,j,t}$ are reduced gradually because of the learning effect, according to the following formula:

$$p_{i,j,t} = \frac{p_{i,j,t-1}}{(t+1)^{\alpha}}, \tag{6}$$

where α represents the learning rate and is set to 0.35, as in [39–41]. Owing to the learning effect, $\lim_{t \longrightarrow \infty} p_{i,j,t} = 0$ and $p_{i,j,t} < p^*_{i,j,t} \leq 1$ for all $t^* < t < \infty$ and $0 < p_{i,j,t}$, in accordance with Equation (6). Note that $0 \leq p_{i,j,t} = p^*_{i,j,t} \leq 1$ for all t^* and t if there is no learning effect.

4. Temporal Learning-Effect Spread Probability and Period

Before the proposed method is solved for calculating the probability of a specified number of infected computers conducted from a computer virus during a specific time period, we need to know both the temporal learning-effect spread probability and the length of the infected period. These two factors are discussed in this section.

4.1. Proposed Learning-Effect-SIR Model. The SIR model is adopted. As mentioned in Section 3.1, each node can be categorized into susceptible (S), infected (I), or removed (R) in the SIR model. Each node undergoes the transition of $S \longrightarrow I \longrightarrow R$, that is, from a susceptible node (S) to an infected node (I) and then to a removed node (R) after a certain infected period.

The spread probability $p_{i,j}$ is the probability that the computer virus is spread out from an infected node $i \in V$ to a susceptible node $j \in V\ (i)$. The temporal learning-effect spread probability Pr (node j infected from node i at time t only) = pi,j,t is a special spread probability at timeslot t only, and it varies with the timeslot because of the learning effect of which users know how to resist the computer virus to reduce the loss gradually.

The computer virus can spread to any susceptible node in $V\ (i)$ at timeslot t with probability $p_{i,j,t}$ if node $i \in V$ is infected. The computer virus cannot spread from node $i \in V$ to any neighboring node $j \in V\ (i)$, that is, $p_{i,j,t} = 0$, if i is a removed node. Moreover because of the learning effect, the computer virus can occasionally be detected and killed easily, such that the temporal learning-effect spread probability $p_{i,j,t}$ is decreased if t is increased.

4.2. Initial Spread Probability. $p_{i,j,0} = p_{i,j}$ represents the initial spread probability of node i infected at time 0, where no learning effect is considered, where for all infected node $i \in V$ and susceptible node $j \in V\ (i)$. Hence, the initial spread probability is simple to calculate, and it is derived here before we determine the temporal learning-effect spread probability.

The value of $p_{i,j}$ for all nodes $i \in V$ to $j \in V\ (i)$ is defined according to the PageRank algorithm. Details are presented in Section 3.2, as follows:

$$\Pr_{(ei,j)} = p_{i,j,0} = p_{i,j} == \frac{\mathrm{PR}(j)}{\sum_{k \in V(i)} \mathrm{PR}(k)}. \tag{7}$$

Equation (7) is based on the fundamental concept of the scale-free network: a larger PageRank number, that is, a higher node distribution probability, corresponds to a higher spread out probability, that is, $p_{i,j}$ is proportional to PR (j). For example, suppose that after node 0 is infected, we have $p_{0,5}$ and $p_{0,7}$ based on $V\ (0) = \{5, 7\}$ and equation (7):

$$p_{0,5} = \frac{0.138452}{0.138452 + 0.132599} = 0.510797, \tag{8}$$

$$p_{0,7} = \frac{0.132599}{0.138452 + 0.132599} = 0.489203. \tag{9}$$

According to equations (8) and (9), we have four possible situations: infected node 0 can spread the computer virus to susceptible node 5 only, node 7 only, both nodes 5 and 7, or nowhere with the following probabilities:

$$\begin{aligned} p_{0,5}p_{0,7} &= 0.249883, \\ p_{0,5}\left(1 - p_{0,7}\right) &= 0.249883, \\ \left(1 - p_{0,5}\right)p_{0,7} &= 0.239320, \\ \left(1 - p_{0,5}\right)\left(1 - p_{0,7}\right) &= 0.249883, \end{aligned} \tag{10}$$

respectively.

Without considering the learning effect, the initial spread probability of each infected node is provided below according to the adjacent matrix and PagePank values listed in Tables 3 and 5, respectively.

4.3. LAGS. The infected timeslot of node $i \in V$ is denoted as t_i if node i is infected at time t_i. Moreover, the infected timeslot of any node spread from node i can only be after or equal to t_i. If node i is infected at time t_i but only starts to spread to node j at time t_j, there is a lag of $\Delta t_{i,j} = t_i - t_j$. The value of $\Delta t_{i,j}$ can be any nonnegative integer, and it is called a no-lag infection if $\Delta t_{i,j} = 0$.

For example, in Figure 2, let the computer virus start spreading after infecting node 0 at time 0, that is, $t_0 = 0$. Assuming that susceptible node 5 is infected from node 0 and susceptible node 7 is infected from node 5, we have $t_0 = 0 \le t_5$ and $t_5 \le t_7$, for example, $t_5 = t_7 = \Delta t_{0,5} = \Delta t_{5,7} = 0$; $t_5 = \Delta t_{0,5} = 1$, $t_7 = 2$, and $\Delta t_{5,7} = 1$; $t_5 = \Delta t_{0,5} = 2$, $t_7 = 3$, and $\Delta t_{5,7} = 1$. Moreover, as shown in Table 6, the whole network is infected at timeslot 2 if node 0 spreads the computer virus out to all its neighboring nodes, that is, $V\ (0) = \{5, 7\}$ at timeslot 1, and the computer virus spreads from node 5 to all the nodes connected to node 5, that is, $V\ (5) = \{1, 2, 3, 4, 6\}$, at timeslot 2.

Suppose that all infections have 1-lag as shown in Table 7 and the computer virus is initialized at node 0. In the worst case, the computer virus is spread out in the order of the node labels, that is, 1, 2, . . ., $|V| - 1$, at time 1, 2, . . ., $|V| - 1$, respectively. Hence, we have the following property:

Property 1. The upper bound of the spread period is $|V| - 1$ if all infections are 1-lag infections.

4.4. Spread Probability with Learning Effect. As mentioned in Section 3.4, $p_{i,j,t}$ is reduced according to the learning effect modeled in the following equation:

$$p_{i,j,t} = \frac{p_{i,j,t-1}}{(t+1)^{\alpha}}, \tag{11}$$

For example, the values of $p_{i,j,1}$, $p_{i,j,2}$, and $p_{i,j,3}$ are presented in Tables 8–10 on the basis of the initial spread probability given in Table 11 and Equation (7).

If node i infected at timeslot t did not spread the computer virus at timeslots t, t+1, . . ., and t^*−1 and spread it to node j at timeslot t, the related probability is denoted as $P_{i,j,t}$ and calculated using Equation (12).

$$P_{i,j,t} = \prod_{\tau=0}^{t-1} \prod_{\forall k \in V(i)} \left(1 - p_{i,k,\tau}\right) \times p_{i,j,(t-1)}, \tag{12}$$

Here,

$$\prod_{\forall k \in V(i)} \left(1 - p_{i,k,\tau}\right), \tag{13}$$

and

(i) Input: The number of coordinates n.
(ii) Output: All n-tuple binary-state vectors X.
(iii) STEP B0. Let X be a zero vector, SUM = 0, and $i = 1$.
(iv) STEP B1. If $X_i = 1$, let $X_i = 0$, SUM = SUM −1, and go to STEP B4.
(v) STEP B2. Let $X_i = 1$ and SUM = SUM +1.
(vi) STEP B3. Let $i = i + 1$ and go back to STEP B1 if $i < n$.
(vii) STEP B4. If SUM = n, halt; otherwise, let $i = 1$ and go back to STEP B1.

Algorithm 2: Algorithm BAT.

Table 6: All state vectors generated from (0, 0, 0, 0, 0) using the Bat.

i	X
1	(0, 0, 0, 0, 0) = 00000
2	(0, 0, 0, 0, 1) = 00001
3	(0, 0, 0, 1, 0) = 00010
4	(0, 0, 0, 1, 1) = 00011
5	(0, 0, 1, 0, 0) = 00100
6	(0, 0, 1, 0, 1) = 00101
7	(0, 0, 1, 1, 0) = 00110
8	(0, 0, 1, 1, 1) = 00111
9	(0, 1, 0, 0, 0) = 01000
10	(0, 1, 0, 0, 1) = 01001
11	(0, 1, 0, 1, 0) = 01010
12	(0, 1, 0, 1, 1) = 01011
13	(0, 1, 1, 0, 0) = 01100
14	(0, 1, 1, 0, 1) = 01101
15	(0, 1, 1, 1, 0) = 01110
16	(0, 1, 1, 1, 1) = 01111
17	(1, 0, 0, 0, 0) = 10000
18	(1, 0, 0, 0, 1) = 10001
19	(1, 0, 0, 1, 0) = 10010
20	(1, 0, 0, 1, 1) = 10011
21	(1, 0, 1, 0, 0) = 10100
22	(1, 0, 1, 0, 1) = 10101
23	(1, 0, 1, 1, 0) = 10110
24	(1, 0, 1, 1, 1) = 10111
25	(1, 1, 0, 0, 0) = 11000
26	(1, 1, 0, 0, 1) = 11001
27	(1, 1, 0, 1, 0) = 11010
28	(1, 1, 0, 1, 1) = 11011
29	(1, 1, 1, 0, 0) = 11100
30	(1, 1, 1, 0, 1) = 11101
31	(1, 1, 1, 1, 0) = 11110
32	(1, 1, 1, 1, 1) = 11111

$$\prod_{\tau=0}^{t-1} \prod_{\forall k \in V(i)} \left(1 - p_{i,k,\tau}\right), \tag{14}$$

represent the probabilities that node i did not spread to any node during time τ and any time before t, respectively.

From equation (12), a larger $(t^* - t)$ corresponds to a lower probability of the computer virus spreading from node i regardless of whether there is a learning effect.

For example, node 0 is infected at the beginning in Figure 2. The temporal learning effect spread probabilities of nodes 5, 7, and ∅ are presented in the 2nd, 3rd, and 4th columns below. In addition, the probability that node i spreads the virus to nodes $j = 5$ and 7 are presented in the last two columns. The temporal learning-effect probability $p_{0,5,t}$ is reduced from 0.1610000 at $t = 0$ to 0.1263180, 0.1096058, ..., 0.0719161 at timeslots $t = 1, 2, \ldots, 9$, respectively, as indicated by Table 12. The probabilities that node 0, that is, $P_{0,5,t}$, start to spread to node 5 are 0.088903, 0.046458, 0.024074, ..., 0.000299 at timeslots $t = 1, 2, \ldots, 9$, respectively.

According to Equations (6) and (12), = 0 with the learning effect and $\lim_{t \longrightarrow \infty} P_{i,j,t} = 0$ regardless of whether there is a learning effect, respectively. Hence, the spread stops at a specific timeslot. The infected period is the total time in which either the whole network is infected or the computer virus is removed from the whole network. The node infected period of node i is defined as the time from the infection of the node to any of the following circumstances:

(1) Node i becomes a recovered node, that is, the computer virus-infected node i is killed;
(2) There are no susceptible nodes in $V(i)$, that is, all nodes are either infected nodes or recovered nodes in $V(i)$.

Because of the instantaneous infection, the smallest t such that $p_{i,j,t} = 0$ for all $j \in V(i)$ is called the infected period of node i. $P_{i,j,t} = 0$ if $p_{i,j,t} = 0$ for all $j \in V(i)$.

5. State Vectors

A state vector is a feasible vector to indicate where and when the computer virus spreads. This section proposes a dual-vector form to construct the state vector by integrating the spread vector and the temporal vector, where the former and the latter indicate where and when the computer virus spreads, respectively.

5.1. Spread Vectors. The spread vector is a $|V|$-tuple vector, and the k^{th} coordinate is the state of the node $(k-1)$ or node k if the first node is labeled 0 or 1, respectively. Moreover, in the spread vector, the first infected node is the node where the value of its related coordinate is equal to itself. Let both the first node and the first coordinate be labeled as 0. For example, in Figure 2, the spread vector $X_1 = (0, 5, 5, 5, 7, 0, 2, 0)$ indicates that node 0 is the first infected node and that the virus spreads to nodes 5 and 7; node 5 spreads the virus to nodes 1–3 after it is infected, because the values in coordinates 1–3 are all 5; nodes 4–7 are infected by nodes 7, 0, 2, and 0, respectively.

A spread vector must be feasible; i.e., the spread of the computer virus must be possible. Only nodes in $V(i)$ can spread the virus to or from node $i \in V$. Hence, we have the following important property and such property is implemented in the proposed BAT to have all spread vectors without needing to verify its feasibility to reduce the runtime.

Property 2. Let $X(i)$ be the value in the i^{th} coordinate represented by node i of vector X. A vector X is a feasible spread vector if $X(i) \in V(i)$ for all nodes $i \in V$.

For example, in Figure 2, $X = (0, 5, 5, 5, 7, 0, 7, 0)$ is an infeasible spread vector, because it is impossible for the computer virus to spread from node 7 to node 6; that is, $X(6) = 7 \in V(6)$.

To simplify the use of the proposed BAT, each spread vector is reconstructed and called the labeled spread vector such that.

(1) The value, i.e., j, at coordinate i is the j^{th} node in $V(i)$, of which all nodes are arranged in the increasing order of the node labels.

(2) The first infect node is in bold.

For example, $X_1 = (0, 5, 5, 5, 7, 0, 2, 0)$ discussed above is rewritten as $X^* = (\mathbf{0}, 0, 0, 0, 2, 0, 1, 0)$ because node 0 is the first infected node and must be written in bold, and nodes 5, 5, 5, 7, 0, 2, 0 at coordinates 1–7 are the nodes labeled 0, 0, 0, 2, 0, 1, and 0 in $V(1) = \{5, 7\}$, $V(2) = \{5, 6, 7\}$, $V(3) = \{5, 6\}$, $V(4) = \{5, 6, 7\}$, $V(5) = \{0, 1, 2, 3, 4, 6\}$, $V(6) = \{1, 2, 3, 4, 5, 7\}$, and $V(7) = \{0, 1, 2, 3, 4, 6\}$, respectively.

To clarify, the following list contains the first 10 spread vectors, labeled spread vectors, and their corresponding 1-lag temporal vectors, which are discussed in Section 5.2.

5.2. Basic Temporal Vectors and Instantaneous Infection. A temporal vector is a vector in which the coordinate value is the timeslot of the related node that has the infection. Similar to the labeled spread vector, the first infect node is in bold in the temporal vector. For example, $T = (\mathbf{0}, 2, 2, 2, 2, 1, 2, 1)$ is the timeslot vector with respect to that in Table 13. In T, nodes 0–7 are infected at timeslots 0, 2, 2, 2, 2, 1, 2, and 1, respectively. Moreover, from T, we observe that node 0 is the first infected node, because $T_0 = 0$ and all the infections have 1-lag, as the gap between two consecutive distinctive numbers in T is 1, e.g., 0 and 1; 1 and 2.

Let $T_{i,t}$ be the t-lag temporal vector of node i if all infections are t-lag, that is, $\Delta t_{j,k} = t$ for all $j \in V$ and $k \in V(j)$. Because all infections are t-lag, the following property holds.

Property 3. The t-lag timeslot vector is the upper-bound of any feasible timeslot that has at most t-lag.

We have the following important property that is implemented in the proposed BAT to serve as an upper-bound to help in searching for all possible feasible timeslot vectors.

Property 4. The t-lag timeslot vector $T_{i,t} = t \times T_{i,1}$ for all $i \in V$.

Each infected node can spread the computer virus only after it is infected and before it is cured. The maximal spread period can be infinity if the computer virus is not detected, as discussed in Section 4.1, theoretically. Hence, a new concept called instantaneous infection is provided for defensive pessimism.

In an instantaneous infection, the computer virus can spread without waiting for another timeslot, that is, $\Delta t_{i,j} = 0$ for all $i \in V$ and all j in $V(i)$ if the node is first infected. Note that a 0-lag timeslot $T_{i,0}$ corresponds to a zero vector and an instantaneous infection. Hence, we have the following property:

Property 5. The 1-lag basic temporal vector is the upper-bound of any feasible timeslot vector.

All temporal vectors can be generated according to Property 5 such that each of their coordinates is less than or equal to that of the related basic temporal vector. Moreover, owing to the characteristic of the instantaneous infection, we have the following important property in filtering out the feasible temporal vectors from all vectors generated according to Property 5.

Property 6. The temporal vector Y is feasible if and only if the following conditions are satisfied, where T represents the 1-lag basic temporal vector that generates Y; that is, $Y(i) \le T(i)$ for all node i in V.

(1) $Y(i) < Y(j)$ if $T(i) < T(j)$, where $i, j \in V$,

(2) $Y(i) \le Y(j)$ if $T(i) = T(j)$, where $i, j \in V$.

5.3. Basic State Vectors. The state vector is a dual vector formed by two vectors—a spread vector and a temporal vector—separated using the notation ";". For example, $X_1 = (0, 5, 5, 5, 7, 0, 2, 0; 0, 2, 2, 3, 2, 1, 5, 1)$ indicates the following:

(1) Node 0 is the first infected node at timeslot 0 because it is characterized by $X_1(0) = 0$

(2) Node 0 spreads the virus to nodes 5 and 7 at $t = 1$

(3) Node 5 spreads the virus to nodes 1 and 2 at $t = 2$ and to node 3 at $t = 3$

(4) Node 7 spreads the virus to node 4 at $t = 2$

(5) Node 2 spreads the virus to node 6 at $t = 5$

A state vector is a basic state vector if its temporal vector is basic. A state vector and/or a basic state vector are infeasible if it is impossible to spread the computer virus according to either its spread vector or the temporal vector, that is, $j_{i,t}$, and $j \notin V(i)$ or $t_j \le t_i$; otherwise, it is a feasible state vector. For example, X_1 discussed above is a feasible state vector, $X_2 = (0, 0, 5, 5, 7, 1, 5, 0; 0, 2, 2, 3, 2, 1, 5, 1)$ is infeasible because $1 \notin V(0)$ in its spread vector, and $X_3 = (0, 5, 5, 5, 7, 0, 2, 0; 0, 2, 0, 3, 2, 1, 5, 1)$ is infeasible because $T_2 = 0$ but $T_5 = 1$ and node 2 is infected from node 5.

For any state vector, the feasibility of its temporal vector depends on its spread vector. Moreover, from Property 5, all temporal vectors can be deduced from the 1-lag temporal vectors. Hence, in the proposed BAT, all the feasible spread vectors together with their 1-lag temporal vectors are found

Table 7: Spread period example with 1-Lag.

t	i	Susceptible nodes	Infectious nodes at $(t+1)$
0	0	{5, 7}	{0, 5, 7}
1	5	{1, 2, 3, 4, 6}	{0, 1, 2, 3, 4, 5, 6, 7}

first to reduce the computational burden of searching for all the state vectors.

The following property involves the relationship between the damping factor d and state vectors. According to this important property, we can simply find all the state vectors for a specific d (without finding all the state vectors for any d), which is very useful for improving the efficiency of the related algorithms.

Property 7. Regardless of the values of the damping factor d, a feasible state vector is always feasible for all d.

6. Proposed BAT

Similar to the DFS, BFS, and UGFM, the BAT is an implicit enumeration search method that can find all the feasible state vectors. However, the BAT is easier to code, more flexible to modify, and more efficient in execution than the other methods [40, 50, 51]. Hence, the BAT is adopted in this study and developed in this section formally to solve the proposed problem.

6.1. Basic Idea behind the Proposed BAT. Let the update procedure be started from the last coordinate to the first coordinate. The basic idea in the proposed BAT is redefined in terms of the fundamental concept in the traditional BAT update procedure by changing a binary vector to a multistate vector as follows:

(1) If it is possible to replace the value of the current coordinate with a larger feasible value, it is replaced, and the new vector is a new state vector. For example, in the traditional BAT, the last 0 in the binary-state vector $X_i = (0, 1, 1, 0)$ can be updated to 1, and $X_{i+1} = (0, 1, 1, 1)$ is a new binary-state vector.

(2) If it is impossible to replace the value of the current coordinate with a larger feasible value, the current value is reset to the smallest feasible value, the algorithm moves to the next coordinate, the foregoing procedure is repeated until the replacement is possible, and then the first step is performed. For the example used in the first step, $X_{i+1} = (0, 1, 1, 1)$ is updated to $X_{i+2} = (1, 0, 0, 0)$.

To deal with the temporal and learning effect properties in the proposed BAT, the details of the foregoing new idea are explained in the remainder of this section.

6.2. BAT-1 for Spread Vector and 1-LAG Temporal Vector. There are two BATs in the proposed BAT. The first one, which is called BAT-1, finds all the feasible spread vectors together with the related 1-lag temporal vectors. The second BAT, which is called BAT-2, finds all the feasible temporal vectors according to the found 1-lag temporal vectors.

BAT-1 is proposed here by changing the binary states to multistates to find all the feasible labeled spread vectors to fit the proposed problem, and its pseudocode is presented below (Algorithms 3 and 4):

The above procedure essentially follows the concepts proposed in Section 6.1. For example, according to the proposed BAT-1 algorithm, we have the first 10 spread states and the related temporal vectors, as shown in Table 11 listed in Section 5.1.

6.3. BAT-2 for All Temporal Vectors. From Property 5, all temporal vectors can be obtained from 1-lag temporal vectors. Hence, another BAT based on Properties 3–6 is implemented to find all temporal vectors for each 1-lag temporal vector as follows (Algorithm 5:

For example, there are 2591 1-lag temporal vector candidates generated from the basic 1-lag temporal vector $T = (0, 2, 2, 2, 2, 1, 3, 3)$ obtained according to the feasible labeled spread vector $X = (\mathbf{0}, 0, 0, 0, 0, 0, 0, 3)$, for which the spread vector is (0, 5, 5, 5, 5, 0, 1, 1) in Figure 2. In total, 479 of them are feasible, and the first 30 temporal vectors are presented in Table 14.

6.4. Pseudocode for the Proposed Algorithm. Assume that the computer virus starts to spread from node TARGET, and we wish to determine the probability that the whole network is infected within a t-lag. The pseudocode of the proposed method for solving the foregoing problem, according to the temporal spread probability with the learning effect derived in Section 4.4, the novel dual state vectors developed in Section 5.3, and the new BAT proposed in Sections 6.2 and 6.3, in estimating the infected probability of computer virus spread areas is presented below (Algorithm 6:

For example, let the damping factor be $d = 0.1$, the lag be one timeslot, and node 0 be the first infected node in Figure 2. Then, we can obtain the PageRank values of each node, the degree of each node, the initial spread probability of each directed arc, and the temporal learning-effect spread probability of each directed, as shown in Tables 5, 7, and 9, respectively.

Using the proposed BAT-1 and BAT-2 algorithms, the basic state vectors and state vectors are obtained. The first 10 basic state vectors and the first 30 1-lag temporal vectors generated from the basic 1-lag temporal vector $T = (0, 2, 2, 2, 2, 1, 3, 3)$ are presented in Table 14.

In the last step, that is, STEP 3, the probabilities of all the feasible state vectors are calculated and summed. The final probability of node 0, which is infected first and spreads the virus throughout the whole network within one timeslot, is 0.7038000, and there are 1268 feasible basic state vectors among the 9720 state vector candidates. In addition, the probability of the labeled spread vector $X = (\mathbf{0}, 0, 0, 0, 0, 0, 0,$ and 3), for which the basic 1-lag temporal vector is $T = (0, 2, 2, 2, 2, 1, 3, 3)$, is 1.00259E-08, with 479 feasible 1-lag temporal vectors filtered out from 2591 1-lag temporal vectors, and the probabilities of the first 30 1-lag temporal vectors are presented in Table 15.

Table 8: Spread probability of each infected node at timeslot 1.

	0	1	2	3	4	5	6	7
0						0.40076		0.38382
1						0.26733	0.26122	0.25603
2						0.39683	0.38775	
3						0.26733	0.26122	0.25603
4						0.26733	0.26122	0.25603
5	0.12642	0.12885	0.12600	0.12885	0.12885			
6		0.12569	0.12291	0.12569	0.12569	0.14537		
7	0.15061	0.15350		0.15350	0.15350		0.17347	

Table 9: Spread probability of each infected node at timeslot 2.

	0	1	2	3	4	5	6	7
0						0.27283		0.26130
1						0.18200	0.17783	0.17430
2						0.27016	0.26397	
3						0.18200	0.17783	0.17430
4						0.18200	0.17783	0.17430
5	0.08607	0.08772	0.08578	0.08772	0.08772		0.09913	
6		0.08557	0.08367	0.08557	0.08557	0.09896		0.09478
7	0.10253	0.10450		0.10450	0.10450		0.11809	

Table 10: Spread probability of each infected node at timeslot 3.

	0	1	2	3	4	5	6	7
0						0.16795		0.16085
1						0.11203	0.10947	0.10730
2						0.16630	0.16250	
3						0.11203	0.10947	0.10730
4						0.11203	0.10947	0.10730
5	0.05298	0.05400	0.05280	0.05400	0.05400		0.06102	
6		0.05267	0.05151	0.05267	0.05267	0.06092		0.05834
7	0.06312	0.06433		0.06433	0.06433		0.07269	

Table 11: Initial spread probability of each infected node.

	0	1	2	3	4	5	6	7
0						0.51080		0.48920
1						0.34073	0.33294	0.32633
2						0.50579	0.49421	
3						0.34073	0.33294	0.32633
4						0.34073	0.33294	0.32633
5	0.16114	0.16423	0.16059	0.16423	0.16423		0.18559	
6		0.16021	0.15666	0.16021	0.16021	0.18528		0.17745
7	0.19196	0.19565		0.19565	0.19565		0.22109	

From the above, the proposed BAT can search for complete state vectors that satisfy the requirements of the proposed problem using BAT-1 and BAT-2. Only with all the state vectors can the analytical probability of the proposed algorithm be calculated, as described by STEP 3.

6.5. Experimental Analysis. In a scale-free network, the node degree distribution follows the power law. Thus, the computational burden increases with the size of the free-scale network, according to the power law. To confirm the performance of the proposed BAT in calculating the probability of the entire network being infected by a computer-virus, the proposed algorithm was tested on the mid-size network shown in Figure 2, which datasets are shown on the adjacency matrix in Table 3, by letting each node be the infected node individually for the damping factors of $d = 0.1$, 0.3, 0.5, 0.7, and 0.9 under allowed time lags of $t = 0$, 1, and 2. Hence, there were $8 \times 5 \times 3 = 120$ tests in total.

Table 12: Related probabilities for node 0 in Figure 2.

t	$p_{0,5,t}$	$p_{0,7,t}$	$\prod_{\tau=0}^{t-1}[(1-p_{0,5,\tau})\cdot(1-p_{0,5,\tau})]$	$P_{0,5,t}$	$P_{0,7,t}$
0	0.1610000	0.1352000	0.7038000		
1	0.1263180	0.1060758	0.5402412	0.088903	0.074656
2	0.0859949	0.0722144	0.4547700	0.046458	0.039013
3	0.0529361	0.0444532	0.4104803	0.024074	0.020216
4	0.0301379	0.0253083	0.3877207	0.012371	0.010389
5	0.0160975	0.0135179	0.3762382	0.006241	0.005241
6	0.0081466	0.0068411	0.3705993	0.003065	0.002574
7	0.0039345	0.0033040	0.3679167	0.001458	0.001224
8	0.0018235	0.0015313	0.3666824	0.000671	0.000563
9	0.0008145	0.0006840	0.3661329	0.000299	0.000251

Table 13: Labeled spread vectors and 1-LAG temporal vectors.

	Spread vectors	Labeled spread vectors	1-Lag temporal vectors
1	(5, 5, 5, 5, 5, 0, 1, 1)	(**0**, 0, 0, 0, 0, 0, 0, 1)	(**0**, 2, 2, 2, 2, 1, 3, 3)
2	(5, 5, 5, 5, 5, 0, 1, 3)	(**0**, 0, 0, 0, 0, 0, 0, 2)	(**0**, 2, 2, 2, 2, 1, 3, 3)
3	(5, 5, 5, 5, 5, 0, 1, 4)	(**0**, 0, 0, 0, 0, 0, 0, 3)	(**0**, 2, 2, 2, 2, 1, 3, 3)
4	(5, 5, 5, 5, 5, 0, 1, 6)	(**0**, 0, 0, 0, 0, 0, 0, 4)	(**0**, 2, 2, 2, 2, 1, 3, 4)
5	(5, 5, 5, 5, 5, 0, 2, 0)	(**0**, 0, 0, 0, 0, 0, 1, 0)	(**0**, 2, 2, 2, 2, 1, 3, 1)
6	(5, 5, 5, 5, 5, 0, 2, 1)	(**0**, 0, 0, 0, 0, 0, 1, 1)	(**0**, 2, 2, 2, 2, 1, 3, 3)
7	(5, 5, 5, 5, 5, 0, 2, 3)	(**0**, 0, 0, 0, 0, 0, 1, 2)	(**0**, 2, 2, 2, 2, 1, 3, 3)
8	(5, 5, 5, 5, 5, 0, 2, 4)	(**0**, 0, 0, 0, 0, 0, 1, 3)	(**0**, 2, 2, 2, 2, 1, 3, 3)
9	(5, 5, 5, 5, 5, 0, 2, 6)	(**0**, 0, 0, 0, 0, 0, 1, 4)	(**0**, 2, 2, 2, 2, 1, 3, 4)
10	(5, 5, 5, 5, 5, 0, 3, 0)	(**0**, 0, 0, 0, 0, 0, 2, 0)	(**0**, 2, 2, 2, 2, 1, 3, 1)

(i) Input: A scale-free network $G(V, E)$ and the computer virus infects node TARGET first.
(ii) Output: All state vectors without duplications.
(iii) STEP S0. Let X be a zero vector with n coordinates represented the node states, vector index k = 1, istop = 1 if TARGET = 0, and istop = 0 if TARGET >0.
(iv) STEP S1. Let coordinate index $i = (n-1)$.
(v) STEP S2. If i = TARGET, let $i = (i-1)$ and go to STEP S3.
(vi) STEP S3. If $X(i) < (W_i - 1)$, let $X(i) = X(i) + 1$, and execute 1-lag_Temporal_Vector (X).
(vii) STEP S4. If X_k is feasible, let $k = k + 1$ and go STEP S1.
(viii) STEP S5. If $i = i_{\text{stop}}$, halt and $X_1, X_2, \ldots, X_k$ are all feasible spread vectors.
(xi) STEP S6. Let $X(i) = 0$, $i = (i-1)$, and go to STEP S2.

Algorithm 3: BAT1.

As the number of state vectors increased, the runtime increased exponentially. Hence, we only discuss t values up to 2.

Both BAT-1 and BAT-2 were coded in Python 3.7.7 and run on Spyder 4.1.3. The 160 tests were conducted on Windows 10 with an Intel Core i7-8650U CPU at 1.90 GHz and 2.11 GHz with 16 GB RAM. The experimental results are presented in Tables 16–18.

The numbers of basic state vector candidates N_s and basic feasible state vectors n_s were only related to the node degrees Deg (i) based on the characteristics of the scale-free network and were unrelated to the values of the damping factors (d) and the allowed time lags (t), for all $i \in V$. Hence, we have the same number of basic and feasible state vectors and probabilities if $V(i) = V(j)$, for all nodes i and j. Thus, we list the values of N_s and n_s in Table 14 and not in the other tables.

The foregoing observation is useful, and, accordingly, we must focus on the nodes without the same neighbors. For example, nodes 1, 3, and 4 all have the same neighbors, that is, $V(1) = V(3) = V(4) = \{5, 6, 7\}$, and we can search for the state vectors and calculate the probability for node 1 because nodes 3 and 4 are identical.

Moreover, a higher degree of the first infected node corresponds to a higher probability of having a smaller number of basic state vector candidates. This is because the total initial spread probabilities from one node to its neighbors are one, and the more neighbors have a higher probability of having a lower spread probability. In addition, the more the neighbors, the smaller the values after using multiplications in Equation (12).

For the same reason as in Table 16, the probabilities that the whole network was infected by node i and node j are equal if $V(i) = V(j)$ in Table 17, for all nodes i and j. For

(i) STEP L0. Let FLAG(V) = false for all V ∈ V, FLAG(TARGET) = true, $t = 1$, $L_0 = \{\text{TARGET}\}$, and $L_1 = \varnothing$.
(ii) STEP L1. Let $T_u = t$, FLAG(u) = true, and $L_t = L_t \cup \{u\}$, where for all $L(X_u) = \text{v}$, for all v ∈ L_{t-1}, and FLAG(u) = false.
(iii) STEP L2. If $L_t = \varnothing$, halt and return the information that X is infeasible.
(iv) STEP L3. If FLAG(v) = true for all v ∈ V, halt and return the information that X is feasible.
(v) STEP L4. Let $t = t + 1$, $L_t = \varnothing$, and go to STEP L1.

Algorithm 4: 1-lag_Temporal_Vector (X).

(i) Input: A 1-lag temporal vector T reordered to T^* in decreasing of the coordinate values.
(ii) Output: All feasible temporal vectors respective to T.
(iii) STEP T0. Let Y be a zero vector with n coordinates represented the node infected time, vector index $k = 1$, $i_{\text{stop}} = 1$ if TARGET = 0, and $i_{\text{stop}} = 0$ if TARGET >0.
(iv) STEP T1. Let coordinate index $i = (n - 1)$.
(v) STEP T2. If $Y(i) < T^*(i)$, let $Y(i) = Y(i) + 1$, and execute 1-lag_Temporal_Vector(X).
(vi) STEP T3. If $Y(u) < Y(\text{v})$ and $T^*(u) < T^*(\text{v})$ for all nodes u and v in V, Y is infeasible and go STEP S1. Otherwise, let $k = k + 1$ and go STEP S1.
(vii) STEP T4. If $i = i_{\text{stop}}$, halt and $Y_1, Y_2, \ldots, Y_k$ are all feasible temporal vectors generated from T.
(viii) STEP T5. Let $X(i) = 0$, $i = (i - 1)$, and go to STEP T2.

Algorithm 5: BAT-2.

Table 14: First 30 1-Lag temporal vectors generated from T = (0, 2, 2, 2, 2, 1, 3, 3).

i	T_i
1	(**0**, 0, 0, 0, 0, 0, 0, 1)
2	(**0**, 0, 0, 0, 0, 0, 0, 2)
3	(**0**, 0, 0, 0, 0, 0, 0, 3)
4	(**0**, 0, 0, 0, 0, 0, 1, 0)
5	(**0**, 0, 0, 0, 0, 0, 1, 1)
6	(**0**, 0, 0, 0, 0, 0, 1, 2)
7	(**0**, 0, 0, 0, 0, 0, 1, 3)
8	(**0**, 0, 0, 0, 0, 0, 2, 0)
9	(**0**, 0, 0, 0, 0, 0, 2, 1)
10	(**0**, 0, 0, 0, 0, 0, 2, 2)
11	(**0**, 0, 0, 0, 0, 0, 2, 3)
12	(**0**, 0, 0, 0, 0, 0, 3, 0)
13	(**0**, 0, 0, 0, 0, 0, 3, 1)
14	(**0**, 0, 0, 0, 0, 0, 3, 2)
15	(**0**, 0, 0, 0, 0, 0, 3, 3)
16	(**0**, 0, 0, 0, 1, 0, 1, 1)
17	(**0**, 0, 0, 0, 1, 0, 1, 2)
18	(**0**, 0, 0, 0, 1, 0, 1, 3)
19	(**0**, 0, 0, 0, 1, 0, 2, 1)
20	(**0**, 0, 0, 0, 1, 0, 2, 2)
21	(**0**, 0, 0, 0, 1, 0, 2, 3)
22	(**0**, 0, 0, 0, 1, 0, 3, 1)
23	(**0**, 0, 0, 0, 1, 0, 3, 2)
24	(**0**, 0, 0, 0, 1, 0, 3, 3)
25	(**0**, 0, 0, 0, 2, 0, 2, 2)
26	(**0**, 0, 0, 0, 2, 0, 2, 3)
27	(**0**, 0, 0, 0, 2, 0, 3, 2)
28	(**0**, 0, 0, 0, 2, 0, 3, 3)
29	(**0**, 0, 0, 1, 0, 0, 1, 1)
30	(**0**, 0, 0, 1, 0, 0, 1, 2)

example, $V(1) = V(3) = V(4) = \{5, 6, 7\}$ from Table 5; the probabilities that the whole network was infected by nodes 1, 3, and 4 were 1.78550E-05 for $d = 0.1$ and $t = 0$.

The increment in d reduced the obtained probability. For example, for $t = 0$ and $i = 0$, the related probability was 1.05186E-05 for $d = 0.1$, and it was higher than 7.55361E-06 for $d = 0.3$, as shown in Table 15. This is because a smaller d corresponded to a higher probability of exploring new areas.

A virus that spent a longer amount of time spreading had a higher probability of spreading to the entire network. Hence, the increment of t increased the obtained probability; for example, the probability was 1.05186E-05 for $t = 0$ and increased to 1.64706E-05 for $t = 1$, as shown in Table 17. However, the rate of the increment in the probability decreased as t increased. This is because a larger value of t corresponded to a smaller number of susceptible nodes remaining, which resulted in a lower spread probability. For example, 1.64706E-05/1.05186E-05 = 1.56585477 > 1.67060 E-05/1.64706E-05 = 1.014292133, where 1.05186E-05, 1.64706E-05, and 1.67060E-05 represent the spread probabilities of $t = 0$, 1, and 2 for $i = 0$ and $d = 0.10$, respectively.

Similar to the foregoing observations in Table 17, Table 18 shows that the increment of the allowed time lag t increased the number of temporal state vector candidates Nt and feasible temporal vectors nt; e.g., Nt was 11651837 for t = 1 and increased to 514419848 for t = 2.

Interestingly, the ratio of (the number of temporal state vector candidates)/(the number of temporal state vectors), that is, N_t/n_t in Table 18, increased with t. This confirms that a larger value of t corresponded to a larger N_t and n_t. In

(i) Input: A scale-free network G (V, E), the first infected node TARGET, the damping factor d, and the allowed timeslot lag t.
(ii) Output: The probability that the computer virus spreads throughout the whole network within time lag t.
(iii) STEP 0. Count the degree of each node, calculate the PageRank values of each node, calculate the initial spread probability of each directed arc using Equation (7), and compute the temporal spread probability with the learning effect for each arc using Equation (12).
(iv) STEP 1. Implement the proposed BAT-1 algorithm to search for all feasible basic state vectors constructed by the spread vectors and the 1-lag temporal vectors.
(v) STEP 2. Implement the proposed BAT-2 algorithm to find all the feasible t-lag temporal vectors according to the 1-lag temporal vectors.
(vi) STEP 3. Calculate and sum the probabilities of all the feasible state vectors using Equation (12).

Algorithm 6: Procedure New-BAT.

Table 15: Probabilities of the first 30 1-LAG temporal vectors based on Table 14.

i	Pr (T_i)
1	4.08805E – 09
2	4.72604E – 09
3	5.04121E – 09
4	6.38271E – 09
5	7.02752E – 09
6	7.33750E – 09
7	7.49064E – 09
8	8.14295E – 09
9	8.45649E – 09
10	8.60723E – 09
11	8.68169E – 09
12	9.00670E – 09
13	9.16292E – 09
14	9.23803E – 09
15	9.27513E – 09
16	9.33043E – 09
17	9.35702E – 09
18	9.37015E – 09
19	9.39705E – 09
20	9.40997E – 09
21	9.41636E – 09
22	9.42976E – 09
23	9.43620E – 09
24	9.43938E – 09
25	9.44145E – 09
26	9.44247E – 09
27	9.44349E – 09
28	9.44400E – 09
29	9.49931E – 09
30	9.52589E – 09

Table 16: The Deg (i), N_s, and n_s.

i	Deg (i)	N_s	n_s
0	2	9720	1268
1	3	6480	1269
2	4	9720	1269
3	3	6480	1269
4	3	6480	1269
5	6	3240	1268
6	6	3240	1269
7	5	3888	1269

∗ N_s and n_s represent the numbers of basic state vector candidates and basic feasible state vectors, respectively.

Table 17: Probability that whole network was infected by node i at $t = 0, 1, 2, 3$.

t	id	0.10	0.30	0.50	0.70	0.90
0	0	0.105186E – 04	0.0755361E – 04	0.0556822E – 04	0.0417601E – 04	0.0315968E – 04
	1	**0.178550E – 04**	**0.135186E – 04**	**0.107111E – 04**	**0.0879585E – 04**	**0.0743470E – 04**
	2	0.111024E – 04	0.0815902E – 04	0.0614844E – 04	0.0472119E – 04	0.0367313E – 04
	3	**0.178550E – 04**	**0.135186E – 04**	**0.107111E – 04**	**0.0879585E – 04**	**0.0743470E – 04**
	4	**0.178550E – 04**	**0.135186E – 04**	**0.107111E – 04**	**0.0879585E – 04**	**0.0743470E – 04**
	5	0.0471508E – 04	0.0332248E – 04	0.0250894E – 04	0.0199343E – 04	0.0164777E – 04
	6	0.0787540E – 04	0.0576780E – 04	0.0435602E – 04	0.0337804E – 04	0.0267799E – 04
	7	0.0807434E – 04	0.0626301E – 04	0.0503922E – 04	0.0417717E – 04	0.0354717E – 04
	SUM	**0.958508E – 04**	0.716219E – 04	0.557541E – 04	0.448334E – 04	0.370098E – 04
1	0	0.164706E – 04	0.117532E – 04	0.0859807E – 04	0.0639718E – 04	0.0480194E – 04
	1	**0.285183E – 04**	**0.216532E – 04**	**0.171670E – 04**	**0.140823E – 04**	**0.118742E – 04**
	2	0.182460E – 04	0.136883E – 04	0.104863E – 04	0.0815713E – 04	0.0640909E – 04
	3	**0.285183E – 04**	**0.216532E – 04**	**0.171670E – 04**	**0.140823E – 04**	**0.118742E – 04**
	4	**0.285183E – 04**	**0.216532E – 04**	**0.171670E – 04**	**0.140823E – 04**	**0.118742E – 04**
	5	0.0723298E – 04	0.0512550E – 04	0.0388383E – 04	0.0309452E – 04	0.0256476E – 04
	6	0.122174E – 04	0.0907971E – 04	0.0693560E – 04	0.0542159E – 04	0.0431858E – 04
	7	0.119262E – 04	0.0914769E – 04	0.0728086E – 04	0.0597267E – 04	0.0502157E – 04
	SUM	**1.51648E – 04**	1.13754E – 04	0.886856E – 04	0.712898E – 04	0.587384E – 04
2	0	0.167060E – 04	0.119286E – 04	0.0873134E – 04	0.0649922E – 04	0.0487995E – 04
	1	**0.292036E – 04**	**0.222009E – 04**	**0.176183E – 04**	**0.144634E – 04**	**0.122025E – 04**
	2	0.184794E – 04	0.138706E – 04	0.106310E – 04	0.0827304E – 04	0.0650205E – 04
	3	**0.292036E – 04**	**0.222009E – 04**	**0.176183E – 04**	**0.144634E – 04**	**0.122025E – 04**
	4	**0.292036E – 04**	**0.222009E – 04**	**0.176183E – 04**	**0.144634E – 04**	**0.122025E – 04**
	5	0.0734090E – 04	0.0520308E – 04	0.0394432E – 04	0.0314450E – 04	0.0260794E – 04
	6	0.124624E – 04	0.0928453E – 04	0.0710513E – 04	0.0556119E – 04	0.0443307E – 04
	7	0.121382E – 04	0.0932064E – 04	0.0742342E – 04	0.0609150E – 04	0.0512156E – 04
	SUM	**1.54738E – 04**	1.16210E – 04	0.906902E – 04	0.729597E – 04	0.6.01522E – 04

Table 18: Numbers of t-LAG temporal vector candidates N_t and feasible temporal vectors n_t.

it	1			2			3		
	N_t	n_t	N_t/n_t	N_t	n_t	N_t/n_t	N_t	n_t	N_t/n_t
0	11651837	891932	13.1	514419848	24700821	20.8	5886456765	232325951	25.3
1	9196660	761060	12.1	396940282	20584520	19.3	4504323732	191929448	23.5
2	11303860	874525	12.9	496287202	24131654	20.6	5666506068	226652525	25.0
3	9196660	761060	12.1	396940282	20584520	19.3	4504323732	191929448	23.5
4	9196660	761060	12.1	396940282	20584520	19.3	4504323732	191929448	23.5
5	3783941	461678	8.2	142973828	11344701	12.6	1538807781	101716441	15.1
6	4434892	507651	8.7	173501602	12759040	13.6	1894848780	115517012	16.4
7	5354260	597341	9.0	212317522	15380446	13.8	2328937092	140479310	16.6

addition, a larger N_t corresponded to a smaller portion of feasible temporal vectors because all the feasible temporal vectors were selected from the temporal state vector candidates.

Table 18 shows another interesting finding that is coincident with those in Tables 16 and 17: the values of N_t for nodes i and j are identical if $V(i) = V(j)$. For example, $N_t = 9196660$ for nodes 1, 3, and 4 because $V(1) = V(3) = V(4) = \{5, 6, 7\}$. Moreover, the values of n_t for nodes i and j are still the same if $V(i) = V(j)$. Hence, the probability that the entire network is infected, N_s, n_s, N_t, and n_t are fixed for nodes i and j if $V(i) = V(j)$.

Summarize the experimental results and present the following recommendations that we can learn from:

(1) The decrement in the damping factors of d helps to obtain a higher probability of exploring new areas.

(2) The increment in the allowed time lags of t increases the obtained probability.

(3) The increment in the allowed time lags of t increases the number of temporal state vector candidates N_t, feasible temporal vectors n_t, and the ratio of (the number of temporal state vector candidates)/(the number of temporal state vectors), that is, N_t/n_t.

7. Conclusions

The spread of computer viruses not only jeopardizes the security of computer and network systems but also hinders their normal operation. Because of the skepticism of both human and software customers in dealing to virus detection, after it was acknowledged that post-creation, a computer virus' spread is aided by the digital antidote and decreases over time. This is called the learning effect.

The spread of a computer virus can be modeled using a scale-free network in which the node degree distribution follows the power rule. A novel computer-virus spread dynamic model with the learning effect based on the scale-free model is proposed. A simple and straightforward method based on the BAT and a novel concept called the temporal learning-effect spread probability and the dual-vector combined with the spread vector and the temporal vector is proposed for modeling the spread of computer viruses and theoretically predicting the analytical probabilities of all the infected-computer scenarios.

The reliability and performance of the proposed BAT was confirmed on a simulated temporal deterioration-effect scale-free network generated using the Barabási–Albert model. The results encourage the extension of the proposed BAT, including the development of the Monte Carlo [7] to BAT to solve larger-size scale-free problems.

In the future, we will strive for opportunities to collaborate with government and private organizations on this research to further verify how well the model proposed in this study fits in real life. In addition, comparing the current model with a virus-resistant model, with defense mechanism, will be planned for future works.

Moreover, several extended topics will be considered in the future for advanced research. For example, how to adjust the parameters of the model for calculating the probability of spreading will be studied when there are firewall devices, IDS, IPS, and DMZ in the intranet. We will further incorporate the version of the operating system, the status of opening service port, the status of protocol service, and the executive time of facility, etc., in order to define how these factors are influencing the probability of virus spreading.

Also, we will carefully consider whether to check for existing problems to ensure that the optimal control problem is solvable before attempting to formulate a solution. The proposed algorithm results will be compared with other methods provided in the literature, and the simulation results will also be displayed in a graphical format. Indeed, stability is an important issue for all dynamic systems. However, this paper is more academic. In fact, this paper has already considered stability in the learning effect. Moreover, we will discuss stability separately in the future. Later studies will discuss the scalability of the model and use a large network of 3000 to 5000 routers to understand the proposed performance algorithm and to understand the possibility of when the infection is less likely to become endemic or when it is more likely to become endemic.

Data Availability

The datasets are shown on the adjacency matrix in Table 3.

Disclosure

This article was once submitted to arXiv as a temporary submission that was just for reference and did not provide the copyright.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported in part by the Ministry of Science and Technology, R.O.C. under grant MOST 102-2221-E-007-086-MY3 and MOST 104-2221-E-007-061-MY3.

References

[1] W. Zhang, D. Chen, H. Si, N. N. Xiong, and R. T. D. C. M., "RTDCM: A coding preemption collection system for key data prioritization with hierarchical probability exchange mechanism in mobile computing," *IEEE Access*, vol. 8, pp. 4629–4639, Dec. 2019.

[2] W. C. Yeh, "A greedy branch-and-bound inclusion-exclusion algorithm for calculating the exact multi-state network reliability," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 88–93, Mar. 2008.

[3] X. Zhou, P. Wang, Z. Yang, L. Tong, Y. Wang, C. Yang et al., "A manifold learning two-tier beamforming scheme optimizes resource management in massive MIMO networks," *IEEE Access*, vol. 8, pp. 22976–22987, Jan. 2020.

[4] W. C. Yeh, "A new branch-and-bound approach for the n/2/ flowshop/αF+ βCmax flowshop scheduling problem," *Computers & operations researchComputers & Operations Research*, vol. 26, no. 13, pp. 1293–1310, 1999.

[5] W. C. Yeh, C. Bae, and C. L. Huang, "A new cut-based algorithm for the multi-state flow network reliability problem," *Reliability Engineering & System Safety*," vol. 136, pp. 1–7, Apr. 2015.

[6] M. Wu, L. Zhong, L. Tan, and N. Xiong, "The sequential fusion estimation algorithms based on gauss-Newton method over multi-agent networked systems," *The Sequential Fusion Estimation Algorithms Based on Gauss-Newton Method Over Multi-Agent Networked Systems IEEE Access*, vol. 8, pp. 114315–114329, Jun. 2020.

[7] W. C. Yeh, Y. C. Lin, and Y. Y. Chung, "Performance analysis of cellular automata Monte Carlo Simulation for estimating network reliability," *Expert Systems with Applications*, vol. 37, no. 5, pp. 3537–3544, May 2010.

[8] M. Huang, A. Liu, N. N. Xiong, T. Wang, and A. V. Vasilakos, "An effective service-oriented networking management architecture for 5G-enabled internet of things," *Computer Networks*, vol. 173, Article ID 107208, 2020.

[9] W. C. Yeh, "A simple algorithm to search for all MCs in networks," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1694–1705, Nov. 2006.

[10] W. C. Yeh, "An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability," *IEEE*

Transactions on Reliability, vol. 64, no. 4, pp. 1185–1193, Sep. 2015.

[11] W. C. Yeh, "A simple universal generating function method to search for all minimal paths in networks," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 6, pp. 1247–1254, Nov. 2009.

[12] W. C. Yeh, "A simple universal generating function method for estimating the reliability of general multi-state node networks," *IIE Transactions*, vol. 41, no. 1, pp. 3–11, Nov. 2008.

[13] H. W. Corley, J. Rosenberger, W. C. Yeh, and T. K. Sung, "The cosine simplex algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 27, no. 9-10, pp. 1047–1050, Apr. 2005.

[14] C. M. Lai, W. C. Yeh, and C. Y. Chang, "Gene selection using information gain and improved simplified swarm optimization," *Neurocomputing*, vol. 218, pp. 331–338, Dec. 2016.

[15] C. Wu, S. Guo, Y. Wu, J. Ai, and N. N. Xiong, "Networked fault detection of field equipment from monitoring system based on fusing of motion sensing and appearance information," *Networked Fault Detection of Field Equipment from Monitoring System Based on Fusing of Motion Sensing and Appearance Information Multimedia Tools and Applications*, vol. 79, no. 23-24, pp. 16319–16348, Jun. 2020.

[16] W. C. Yeh, "Search for MC in modified networks," *Computers & Operations ResearchComputers & Operations Research*, vol. 28, no. 2, pp. 177–184, Feb. 2001.

[17] J. von Neumann, "Theory of Self-Reproducing Automata," *Essays on Cellular Automata*, pp. 64–87, University of Illinois Press, Champaign, IL, USA, 1966.

[18] Wikipedia,https://en.wikipedia.org/wiki/Computer_virus#Historical_development.

[19] V. Risak, *Selbstreproduzierende Automaten mit minimaler Informationsübertragung*, Zeitschrift für Maschinenbau und Elektrotechnik, Vienna, Austria, 1972.

[20] J. Kraus, "Selbstreproduktion bei Programmen," Thesis, University of Dortmund, Dortmund, Germany, 1980.

[21] S. I. Shoutkov and A. V. Spesivtsev, "Computer viruses: ways of reproduction in MS-DOS," in *Proceedings of the . 25th Annual 1991 IEEE International Carnahan Conference Oon Security Technology*, Taipei, Taiwan, Oct. 1991.

[22] C. Griffin and R. Brooks, "A note on the spread of worms in scale-free networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 1, pp. 198–202, Feb. 2006.

[23] T. N. Witte, "Phantom malware: conceal malicious actions from malware detection techniques by imitating user activity," *IEEE Access*, vol. 8, pp. 164428–164452, Sep. 2020.

[24] S. Peng, S. Yu, and A. Yang, "Smartphone malware and its propagation modeling: a survey," *IEEE Communications Surveys & TutorialsIEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 925–941, 2014.

[25] P. Li and X. Yang, "On dynamic recovery of cloud storage system under advanced persistent threats," *IEEE Access*, vol. 7, pp. 103556–103569, Jul. 2019.

[26] W.W Tian, X. Ji, W. Liu, G. Liu, J. Zhai, Y. Dai et al., "Prospect theoretic study of honeypot defense against advanced persistent threats in power grid," *IEEE Access*, vol. 8, pp. 64075–64085, Apr. 2020.

[27] J. Singh, D. Kumar, Z. Hammouch, and A. Atangana, "A fractional epidemiological model for computer viruses pertaining to a new fractional derivative," *Applied Mathematics and Computation*, vol. 3161316, pp. 504–515, January 2018.

[28] X. Wang, W. Ni, K. Zheng, R. P.R.P. Liu, and X. Niu, "Virus propagation modeling and convergence analysis in large-scale networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2241–2254, Oct. 2016.

[29] Y. Huang and Q. Zhu, "A differential game approach to decentralized virus-resistant weight Adaptation policy over complex networks," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 2, pp. 944–955, June 2020.

[30] M. W. Eichin and J. A. Rochlis, "With microscope and tweezers: an analysis of the internet virus of November 1988," in *Proceedings of the IEEE Symposium on Security a and Privacy*, Oakland, CA, USA, November 1989.

[31] E. H. Spafford, "Computer viruses as artificial life," *Artificial Life*, vol. 1, no. 3, pp. 249–265, April 1994.

[32] R. Davis, "Exploring computer viruses," in *Proceedings of the in Fourth Aerospace Computer Security Applications*, Orlando, FL, USA, September 1988.

[33] E. Okamoto and H. Masumoto, "ID-based authentication system for computer virus detection," *Electronics Letters*, vol. 26, no. 15, pp. 1169-1170, July 1990.

[34] D. Spinellis, "Reliable identification of bounded-length viruses is NP-complete," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 280–284, Jan 2003.

[35] X. Wang, N. L. Or, Z. Lu, and D. Pao, "Hardware aAccelerator to dDetect mMulti-sSegment vVirus pPatterns," *The Computer Journal*, vol. 58, no. 10, pp. 2443–2460, Oct. 2015.

[36] C. Gan, X. Yang, W. Liu, and Q. Zhu, "A propagation model of computer virus with nonlinear vaccination probability," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 1, pp. 92–100, January 2014.

[37] G. M. Al-Dossary, "Computer vvirus pprevention and ccontainment on mmainframes," in *Proceedings of the International Carnahan Conference Oon Security Technology*, Zurich, Switzerland, October 1989.

[38] H. Yuan, J. Wu, and G. Chen, "Infection functions for virus propagation in computer networks: aAn empirical study," *Tsinghua Science and Technology*, vol. 14, no. 5, pp. 669–676, Oct. 2009.

[39] M. Youssef and C. Scoglio, "Optimal nNetwork-bBased iIntervention in the pPresence of uUndetectable vViruses," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1347–1350, Aug. 2014.

[40] W. C. Yeh, "Novel binary-addition tree algorithm (BAT) for binary-state network reliability problem," 2020, http://arxiv.org/abs/2004.08238.

[41] J. C. Lang, H.H. D. SterckDe Sterck, J. L. Kaiser, J. C. Miller, and J. Gleeson, "Analytic models for SIR disease spread on random spatial networks," *Journal of Complex Networks*, vol. 6, no. 6, pp. 948–970, Dec. 2018.

[42] K. Zhu and L. Ying, "Information source detection in the SIR model: a sample-path-based approach," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 408–421, Feb. 2016.

[43] V. P. Dubey, R. Kumar, and D. Kumar, "A hybrid analytical scheme for the numerical computation of time fractional computer virus propagation model and its stability analysis," *Chaos, Solitons & Fractals*, vol. 133, Article ID 109626, 2020.

[44] J. Ren and Y. Xu, "A compartmental model for computer virus propagation with kill signals," *Physica A: Statistical Mechanics and Iits Applications*, vol. 48615486, pp. 446–454, November 2017.

[45] Q. A. Dang and M. T. Hoang, "Positivity and global stability preserving NSFD schemes for a mixing propagation model of computer viruses," *Journal of Computational and Applied Mathematics*, vol. 37415, Article ID 112753, August 2020.

[46] W. C. Yeh, "Methodology for the reliability evaluation of the novel learning-effect multi-state flow network," *IISE Transactions*, vol. 49, no. 11, pp. 1078–1085, 2017.

[47] W. C. Yeh, P. J. Lai, and M. C. Chuang, "Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects," *Information Sciences*, Information Sciences, vol. 269, pp. 142–158, 2014.

[48] W. C. Lee, M. C. Chuang, and W. C. Yeh, "Uniform parallel-machine scheduling to minimize makespan with position-based learning curves," *Computers & Industrial EngineeringComputers & Industrial Engineering*, vol. 63, no. 4, pp. 813–818, 2012.

[49] W. C. Yeh, "Simplified swarm optimization in disassembly sequencing problems with learning effects," *Computers & Operations Research Computers & Operations Research*, vol. 39, no. 9, pp. 2168–2177, 2012.

[50] Y. Z. Su and W. C. Yeh, "Binary-addition tree algorithm-based resilience assessment for binary-state network problems," *Binary-Addition Tree Algorithm-Based Resilience Assessment for Binary-State Network Problems Electronics*, vol. 9, no. 8, p. 1207, 2020.

[51] W. C. Yeh, "Predicting and modeling wildfire propagation areas with BAT and maximum-state pageRank," *Applied Sciences*, IEEE Access Reviewing Paper, vol. 10, 2020/07.