Computation of Activation Probabilities in the Independent Cascade Model

Wenjing Yang[1], Leonardo Brenner[2] and Alessandro Giua[3]

***Abstract*— Based on the concepts of *word-of-mouth effect* and *viral marketing*, the diffusion of an innovation may be triggered starting from a set of initial users. Estimating the influence spread is a preliminary step to determine a suitable or even optimal set of initial users to reach a given goal. In this paper, we focus on a stochastic model called the Independent Cascade model, and compare a few approaches to compute activation probabilities of nodes in a social network, i.e., the probability that a user adopts the innovation. In the paper, first we propose the *Path Method* which computes the exact value of the activation probabilities but it has high complexity. Second an approximated method, called *SSS-Noself*, is obtained by modification of the existing *SteadyStateSpread* algorithm, based on fixed-point computation, to achieve a better accuracy. Finally an efficient approach, also based on fixed-point computation, is proposed to compute the probability that a node is activated though a path of minimal length from the seed set. This algorithm, called *SSS-Bound-t* algorithm, can provide a lower-bound for the computation of activation probabilities.**

I. INTRODUCTION

In recent years, a large number of social network sites have appeared to connect people and groups together. Networks have been proved to be a good tool to obtain information and communicate ideas. Besides, they are becoming an effective marketing platform, through which it is possible to spread information or products to a large scale with a high speed.

Consider the following marketing example: a company designs a new APP for online users and aims to market it through the social network. It can only choose a small number of users to try the APP initially (because usually a company has limited budget and manpower on a product). Then the company encourages these users to recommend the APP to their friends. And their friends would use it and recommend to their friends and so on. Whether users adopt an innovation is strongly influenced by their acquaintances. That is called the "*word-of-mouth*" effect [9] and this type of marketing is called the *viral marketing* [9] since it is similar to the spread of an epidemic.

The studies on the diffusion of innovations in social networks began in the middle of the 20th century [10], [12]. Motivated by the application of *viral marketing*, Domingos and Richardson [9] proposed a general framework for the application of data mining and modeled the social network as a Markov random field. Kempe et al. [2] proposed two diffusion models, namely the *Independent Cascade model* and the *Linear Threshold model*, to model the propagation of innovations. Besides, they also formulated the issue of choosing influential sets of individuals as a discrete optimization problem. It aims to identify a small subset of initial adopters in a social network to maximize the influence propagation under a given diffusion model. They also proved this *influence maximization* problem is NP-hard and gave a greedy approximation algorithm which guarantees, under certain conditions, that the influence spread is within $(1 - 1/e)$ of the optimal influence spread. However, this approach requires long time to run the simulation, thus later much effort was devoted to derive more efficient algorithms (e.g., [13], [11], [4]).

A preliminary step to determine a suitable or even optimal set of initial users is to compute the final adoption of an innovation from the set of initial users. When the Independent Cascade model is considered, the measure of the influence spread is given by the *activation probability* of an individual, i.e., the probability that the individual adopts the innovation. Monte Carlo simulation proposed by Kempe et al. [2] is a simple and easy way to compute the activation probabilities but it is also quite time-consuming. Aggarwal et al. [6] gave a more efficient approximate algorithm called *SteadyStateSpread*. However, the computed solution may be far from the exact one, depending on the network structure, and there are no guaranteed bounds.

In this paper, focusing on the Independent Cascade model, we analyse the approaches for computing the activation probabilities of individuals. Our main contributions can be summarised as following:

1) To compute the exact solution in small networks, we propose a method that explores all possible evolutions of a model: we call this approach the *Path Method*. We point out that, due to its complexity, this method is only viable for small networks but it is useful to test the correctness of different approaches.
2) We point out two factors leading to the gap between the result of *SteadyStateSpread* and the exact solution: the dependent relation of individuals and the existence of circuits in the net structure. To partially overcome the error caused by circuits, we further propose a new *SSS-Noself* algorithm which updates the activation probability of one node assuming that it has not been activated at all before.

*This work was supported by the China Scholarship Council (CSC)

[1]Wenjing Yang is with Aix-Marseille Université, CNRS, Université de Toulon, LIS UMR 7020, France. wenjing.yang@lis-lab.fr

[2]Leonardo Brenner is with Aix-Marseille Université, CNRS, Université de Toulon, LIS UMR 7020, France. leonardo.brenner@lis-lab.fr

[3]Alessandro Giua is with Aix-Marseille Université, CNRS, Université de Toulon, LIS UMR 7020, France and also with University of Cagliari, DIEE, Italy. alessandro.giua@lis-lab.fr, giua@diee.unica.it

3) We propose an efficient way to compute the activation probabilities along paths of bounded length and provide a lower-bound for the activation probabilities by *SSS-Bound-t*.

The rest of the paper is organized as follows. Section 2 reviews the Independent Cascade model and formally defines the activation probability. Section 3 proposes the *Path Method*. Section 4 analyses *SteadyStateSpread*, further proposes the *SSS-Noself* and *SSS-Bound-t* algorithms. Section 5 presents a series of experimental results. We conclude the paper in Section 6.

II. Background

In this paper, we use the Independent Cascade model to describe the propagation of innovations through social networks. For convenience, the variables used extensively throughout the paper are listed in Table I.

TABLE I
VARIABLE EXPLANATION

Variable	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	a network with node set $\mathcal{V}$ and edge set $\mathcal{E}$
$\mathcal{G}_{IC} = (\mathcal{V}, \mathcal{E}, p)$	an Independent Cascade model
$\mathcal{G}_{IC}^{[q]} = (\mathcal{V}, \mathcal{E}^{[q]}, p^{[q]})$	an Independent Cascade model without node q's influence
N	number of nodes in $\mathcal{G}$
$\mathcal{N}_j^{in}$	set of nodes with direct influence on node j
$\mathcal{N}_j^{out}$	set of nodes on which node j has direct influence
ϕ_0	seed set
$\lvert\phi_0\rvert$	number of nodes in the seed set
π_j	activation probability of node j
π_j^p	activation probability of node j computed by the *Path Method*
π_j^s	activation probability of node j computed by *SteadyStateSpread*
π_j^n	activation probability of node j computed by *SSS-Noself*
$\pi_j^{[q]}$	activation probability of node j without node q's influence
π_j^t	activation probability of node j computed by *SSS-Bound-t*

A. Network Structure

As in most literatures, the social network is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is a set of nodes involved in the network and in this paper we use the terms *individual* or *node* interchangeably. An edge $(i, j) \in \mathcal{E}$ denotes that node i influences node j directly [2].

To denote all individuals with direct influence on node j, we represent the *in-neighbors* of node j as $\mathcal{N}_j^{in} = \{i \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. The *out-neighbors* of node j denoted as $\mathcal{N}_j^{out} = \{i \in \mathcal{V} | (j, i) \in \mathcal{E}\}$ represent the individuals on which node j has direct influence.

B. Independent Cascade Model

The Independent Cascade model [17] is a type of epidemic models, which is based on the assumption that a node may adopt an innovation when one of its in-neighbors has adopted the innovation. In the Independent Cascade model, we add to every edge $(i, j) \in \mathcal{E}$ a *propagation probability* $p : (\mathcal{V} \times \mathcal{V}) \rightarrow (0, 1]$, where $p_{i,j}$ represents the probability that node j is influenced by node i through the edge (i, j) at step k when node i is activated at step $k - 1$. Thus, we denote an Independent Cascade model by a triple $\mathcal{G}_{IC} = (\mathcal{V}, \mathcal{E}, p)$.

Let us define ϕ_0 as the *seed set*, i.e., the set of nodes which have adopted the innovation at step $k = 0$. Then the innovation propagates from the seed set. Each node can be either *active* or *inactive*. When it adopts the innovation (is activated), it becomes active, otherwise is said to be inactive. We also assume that nodes can switch from being inactive to being active, but can not switch in the other direction. It means that the adoption of an innovation is permanent and for this reason the model is called *progressive* [1].

Note that every active node has only one chance to influence each of its out-neighbors. If it fails, it can not try again to activate the same out-neighbor. If there are many in-neighbors of inactive node j that are activated at step $k - 1$, the order in which they attempt to activate node j at step k does not affect the probability of node j being activated. This is called *order-independence*.

C. Activation Probability

We focus on the evaluation of influence propagation through a network. Based on the Independent Cascade model, we analyze different approaches to compute the activation probabilities of nodes.

Definition 1 (Activation Probability): Given an Independent Cascade model $\mathcal{G}_{IC} = (\mathcal{V}, \mathcal{E}, p)$ and a seed set ϕ_0, the probability that a node $j \in \mathcal{V}$ is activated during the dynamic evolution is defined as the activation probability of node j, denoted as π_j.

III. Exact Computation of Activation Probabilities

In this section, we propose an algorithm to compute the exact solution to the influence propagation called the *Path Method*. The method was suggested to us by prof. Chtistoforos Hadjicostis.

The *Path Method* takes into account all evolutions of a model, so that it can offer precise result of the final influence propagation. Firstly, it creates an evolution graph for a model, which is composed of cells shown in Figure 1. Each cell C_k consists of three elements: *past active nodes* is a set A_k^p which contains all the nodes activated before the current step; *current active nodes* is a set A_k^c which contains the nodes activated in the current step; *cell probability* P_k is the probability that the evolution described by A_k^p and A_k^c occurs. The cell whose *current active nodes* is null is denoted as *terminal cell*. Then adding together all the cell probabilities of terminal cells whose *past active nodes* contains node j, the exact activation probability of node j can be computed.

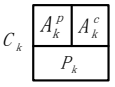


Fig. 1. Cell of evolution graph

For example, the evolution graph for the model in Fig. 2 with seed set $\phi_0 = \{5\}$ is shown in Fig. 3. We briefly explain how to construct the evolution graph of this model.

In C_1, only seed node 5 is activated, thus $A_1^p = \emptyset$ and $A_1^c = \{5\}$. Apparently, the state that only the seed node is activated appears in all evolutions, thus $P_1 = 1$. Since the out-neighbors of node 5 only contains node 3, only C_2 and C_3 can be obtained from C_1: C_2 corresponds to an evolution that does not activate node 3, while C_3 corresponds to an evolution that activates node 3. C_2 is a terminal cell since $A_2^c = \emptyset$. Moreover, $p_{5,3} = 0.4$ and $P_1 = 1$, thus $P_3 = 0.4$. Then four cells can be reached from C_3 according to which subset of out-neighbor of node 3 will be activated. Based on this procedure, the evolution graph corresponding to the network can be obtained. The cell probability of every terminal cell represents the probability of the corresponding evolution. For example, C_{10} corresponds to the evolution where nodes $\{1, 3, 5\}$ are influenced by the order $5 \to 3 \to 1$, and no other node is activated. Thus the terminal cells which contain node j as a past active node describe all final evolutions in which node j can be activated. The sum of the cell probabilities of these terminal cells is the activation probability of node j. For the network in Fig. 2, the activation probabilities obtained from the evolution graph in Fig. 3 are shown in the second row of Table II.

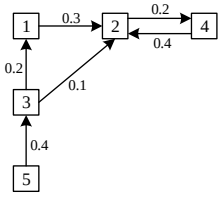


Fig. 2. An Independent Cascade model with 5 nodes

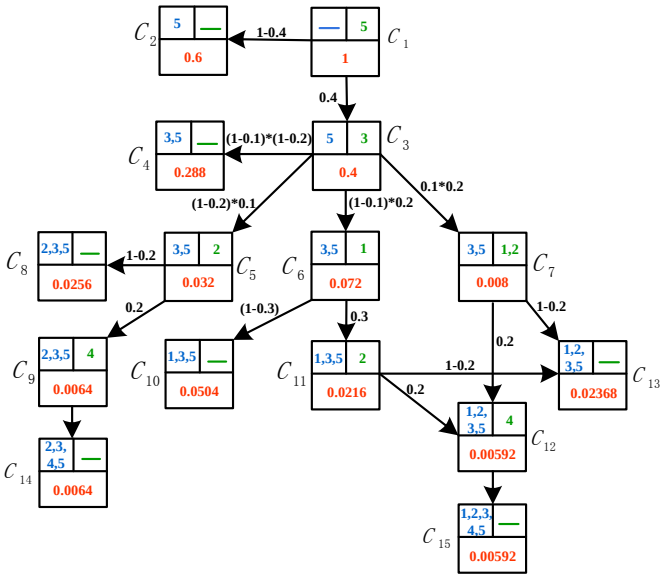


Fig. 3. Evolution representation of the network in Fig. 2

There are three possible states for each node $j \in \mathcal{V}$ in a cell of the evolution graph: belonging to *past active nodes*, belonging to *current active nodes* or in neither of these two sets. Thus the maximal number of cells in an evolution graph is 3^N. We need one pass of $j \in \mathcal{V}$ for each cell to obtain π_j^p. Hence the time complexity of this method is $O(N \cdot 3^N)$. Due to its exponential complexity, this method is only viable for small networks.

IV. Approximate Computation of Activation Probabilities by Fixed-Point Approaches

Aggarwal et al. [6] proposed the *SteadyStateSpread* algorithm to evaluate the steady state assimilation probabilities of all nodes, i.e., the activation probabilities of nodes in this paper. This iterative method computes an approximated value of the node activation probability by solving a non-linear system of equations.

A. Basic Fixed-Point Approach

In this part, we discuss the *SteadyStateSpread* algorithm based on fixed-point theory.

Definition 2 ([14]): Given a real function of a real variable $f : \mathbb{R}$ to $\mathbb{R}$, a real number x is a *fixed point* of f if it satisfies

$$x = f(x)$$

Given a point x_0 in the domain of f, the *fixed-point iteration* is

$$x_{s+1} = f(x_s),\ s = 0, 1, 2, \ldots$$

which generates the sequence $x_0, x_1, x_2, \ldots$ If the sequence converges to a point x and f is continuous, then one can prove that x is a fixed point of f.

To apply this theory for iterative activation probability computation, it is necessary to construct a function for computing π_j^s. According to [6], node j can be activated by either of its in-neighbors. Equivalently, in order for node j to not be activated, it must not be activated by any of its in-neighbors. Assuming that the activation of the in-neighbors are independent events and that they do not depend on the activation of node j, the probability of that can be written as $\prod_{i \in \mathcal{N}_j^{in}} (1 - \pi_i^s \cdot p_{i,j})$. Thus the computation function can be constructed as following:

$$\pi_j^s(k+1) = \begin{cases} 1 & \text{if } j \in \phi_0 \\ 1 - \prod\limits_{i \in \mathcal{N}_j^{in}} (1 - \pi_i^s(k) \cdot p_{i,j}) & \text{if } j \notin \phi_0 \end{cases} \tag{1}$$

As an example, applying *SteadyStateSpread* to the network in Fig. 2, we can compute the activation probabilities shown in the third row of Table II.

B. Improving the Fixed-Point Approach

SteadyStateSpread is generally not correct because Equation 1 holds only assuming that the activation events of node j's in-neighbors are independent events and that these events do not depend on the activation of node j itself. Thus it can not provide exact solution for every structure of networks, especially subgraphs containing bidirectional nodes or dependent sub-structures. Yang et al. [7] discussed the

scenario of *structural defect*, corresponding to this situation: nodes $i, j \notin \phi_0$ and every path from ϕ_0 to j has to pass i, nevertheless according to a certain computation algorithm, π_i depends on π_j. However, they did not involve the whole cases.

Comparing the results computed by the *Path Method* and *SteadyStateSpread* shown in Table II, we find $\pi_2^s > \pi_2^p$ and $\pi_4^s > \pi_4^p$ (which are in bold in Table II). As mentioned in [7], one reason is: in Equation 1, π_2^s depends on π_4^s, i.e., node 4 increases π_2^s. However, node 4 can be activated only after node 2's activation. Another reason is that the dependent relation between node 2's in-neighbors (node 1 and node 3) increases the final result of node 2. Ignoring the influence of node 4, the equation to compute the activation probability of node 2 by *SteadyStateSpread* is

$$\begin{aligned} \pi_2^s &= 1 - (1 - \pi_3^s p_{3,2}) \cdot (1 - \pi_1^s p_{1,2}) \\ &= \pi_3^s p_{3,2} + \pi_1^s p_{1,2} - \pi_1^s \pi_3^s p_{3,2} p_{1,2} \end{aligned} \tag{2}$$

Nevertheless, the exact equation to compute the activation probability of node 2 should be

$$\begin{aligned} \pi_2 &= \pi_3 \cdot (p_{3,2} + (1 - p_{3,2}) \cdot p_{3,1} p_{1,2}) \\ &= \pi_3 p_{3,2} + \pi_1 p_{1,2} - \pi_3 p_{3,2} p_{1,2} p_{3,1} \end{aligned} \tag{3}$$

Moreover, not only the circuit between two nodes such like the one between node 2 and node 4 in Fig. 2, circuits among more than two nodes also contribute to the error in the solution computed by *SteadyStateSpread*.

Overall, circuits and dependent relations among nodes lead to computing activation probabilities by *SteadyStateSpread* that are equal to or greater than the exact solution.

In order to narrow the gap with exact results, Yang et al. [7] proposed the *SSSbyStep* algorithm which incorporates *SteadyStateSpread* with MIP heuristic [3] to limit the iteration time. For partly solving the inaccuracy caused by circuits, we propose in the following an improved algorithm to assure that the iteration process to compute activation probability of node j is not influenced by itself. The new algorithm, that we call *SSS-Noself*, updates activation probabilities of a node by Equation 1 without the influence of the node itself.

Given an Independent Cascade model $\mathcal{G}_{IC} = (\mathcal{V}, \mathcal{E}, p)$, a new network $\mathcal{G}^{[q]} = (\mathcal{V}, \mathcal{E}^{[q]}, p^{[q]})$ is obtained from $\mathcal{G}$ by removing the input and output arcs of node q. The total number of these new nets is N', where $N' = N - |\phi_0|$, $|\phi_0|$ is the number of nodes in the seed set. For node $j \in \mathcal{V}$ one can proceed to compute at each step k the activation probability of j assuming that q has not been activated $\pi_j^{[q]}(k)$. Finally, when updating the activation probability of node j by Equation 1, the used value of every j's in-neighbor i is $\pi_i^{[j]}$ obtained by iteration. Let us define the activation probability vector of network $\mathcal{G}^{[q]}$ $p^{[q]}$ as:

$$p_{i,j}^{[q]} = \begin{cases} 0 & \text{if } q = i \text{ or } q = j \\ p_{i,j} & otherwise \end{cases} \tag{4}$$

Algorithm 1 is a modified version of *SteadyStateSpread* where the activation probability of node j is computed disregarding the influence of itself. The computation result for the network in Fig. 2 by Algorithm 1 is shown in the fourth row of Table II. The results of node 2 and node 4 are in bold in Table II to highlight that they are different from their results by the *Path Method* and *SteadyStateSpread*. It is obviously that the result of *SSS-noself* is closer to the result of the *Path Method* than *SteadyStateSpread*, i.e., *SSS-noself* is more precise than *SteadyStateSpread*. In fact, *SSS-Noself* always gives a result between the result of the *Path Method* and the result of *SteadyStateSpread*.

Different from *SteadyStateSpread*, for all nodes $j \in \mathcal{V}$, *SSS-Noself* not only computes the activation probability of node j at step k, but also the activation probability of node j at step k assuming that node $q \in \mathcal{V} \setminus \phi_0$ remains inactive. Thus the time complexity of *SSS-Noself* is $O(N^2)$, while that of *SteadyStateSpread* is $O(N)$.

Algorithm 1 *SSS-Noself*

Input: An independent cascade network $\mathcal{G}_{IC} = (\mathcal{V}, \mathcal{E}, p)$; a set of initial nodes $\phi_0 \subset \mathcal{V}$; tolerance ε^*
Output: Activation probability π_j^n for all nodes $j \in \mathcal{V}$

$k = 0$
$\varepsilon = \varepsilon^* + 1$
for $q \in \mathcal{V} \setminus \phi_0$ **do**
  $\pi_j^{[q]}(0) = 1, \forall j \in \phi_0$
  $\pi_j^{[q]}(0) = 0, \forall j \in \mathcal{V} \setminus \phi_0$
end for
while $\varepsilon \geq \varepsilon^*$ **do**
  for $q \in \mathcal{V} \setminus \phi_0$ **do**
    for $j \in \mathcal{V}$ **do**
      if $j \in \phi_0$ **then**
        $\pi_j^{[q]}(k+1) = 1$
        $\pi_j(k+1) = 1$
      else
        $\pi_j^{[q]}(k+1) = 1 - \prod_{i \in \mathcal{N}_j^{in}} (1 - p_{i,j}^{[q]} \cdot \pi_i^{[q]}(k))$
        $\pi_j(k+1) = 1 - \prod_{i \in \mathcal{N}_j^{in}} (1 - p_{i,j} \cdot \pi_i^{[j]}(k))$
      end if
    end for
  end for
  $\varepsilon_1 = \sum_{j \notin \phi_0} |\pi_j(k+1) - \pi_j(k)|$
  $\varepsilon_2 = \sum_{j \notin \phi_0} |\pi_j^{[q]}(k+1) - \pi_j^{[q]}(k)|$ $(q \in \mathcal{V} \setminus \phi_0)$
  $\varepsilon = max(\varepsilon_1, \varepsilon_2)$
  $k = k + 1$
end while
return $\pi_j^n = \pi_j(k-1)$

C. Fixed-Point Computation of Activation Probabilities Along Paths of Bounded Length

In this part, we propose an efficient algorithm, called *SSS-Bound-t*, to compute activation probabilities along paths of bounded length by applying Equation 1.

Let us firstly define the length of the shortest path from seed set to a node $j \in \mathcal{V} \setminus \phi_0$ as sp_j and assume each node $j \in \mathcal{V} \setminus \phi_0$ is reachable from the seed set. Kimura et al. [18] proposed SPM and SP1M, where node j can

be activated only at step $k = sp_j$ in SPM, or only at step $k = sp_j$ as well as step $k = sp_j + 1$ in SP1M. Nevertheless, they did not discuss how to compute these probabilities without previously determining the corresponding paths, a procedure that may be computationally expensive. Maximum influence paths are computed by the Dijkstra algorithm in [3] and [7], which has high complexity. We propose an approach based on fixed-point computation that does not require preliminarily computing the shortest path. In our procedure we record sp_j as step k when π_j^t firstly changes from zero to non-zero. Besides, we set the path bound t to compute π_j^t involving not only the shortest paths but also the paths whose length is no greater than $sp_j + t$. The procedure of *SSS-Bound-t* is shown in Algorithm 2. The computation result for the network in Fig. 2 by *SSS-Bound-0* is shown in the fifth row of Table II.

Algorithm 2 *SSS-Bound-t*

Input: An independent cascade network $\mathcal{G}_{IC} = (\mathcal{V}, \mathcal{E}, p)$; a set of initial nodes $\phi_0 \subset \mathcal{V}$; path bound t
Output: Activation probability π_j^t for all nodes $j \in \mathcal{V}$

Initialize $\pi_j^t(0) = 1$, $j \in \phi_0$; $\pi_j^t(0) = 0$, $j \in \mathcal{V} \setminus \phi_0$; $k = 0$; $step_j = inf$, $j \in \mathcal{V} \setminus \phi_0$
$stop = 0$
while $stop = 0$ **do**
  $stop = 1$
  for $j \in \mathcal{V}$ **do**
    $\pi_j^t(k+1) = \pi_j^t(k)$
    if $j \notin \phi_0$ and $k \leq step_j$ **then**
      $\pi_j^t(k+1) = 1 - \prod_{i \in \mathcal{N}_j^{in}} (1 - p_{i,j} \cdot \pi_i^t(k))$
      $stop = 0$
    end if
    if $\pi_j^t(k+1) \neq 0$ and $\pi_j^t(k) = 0$ **then**
      $step = k + 1 + t$
    end if
  end for
  $k = k + 1$
end while
return $\pi_j^t = \pi_j^t(k-1)$

The *SSS-Bound-t* algorithm generalizes SPM and SP1M [18], exploiting the efficient fixed-point computation of *SteadyStateSpread*. It cannot always give a well approximate result since it depends on the seed set as well as the value of t. However, the result of *SSS-Bound-0* can be regarded as a lower-bound for the exact activation probability. Same with *SteadyStateSpread*, the time complexity of *SSS-Bound-t* is $O(N)$. However, in most cases *SSS-Bound-0* stops the iteration before it converges, thus *SSS-Bound-0* is usually less time-consuming than *SteadyStateSpread*.

D. Comparison of Different Fixed-Point Approaches

In fact, the computing activation probabilities by the algorithms above have a fixed relationship.

TABLE II
COMPARISON OF ACTIVATION PROBABILITIES FOR THE NETWORK IN FIG. 2

Node	1	2	3	4	5
Path Method	0.08	**0.0616**	0.4	**0.0123**	1
SteadyStateSpread	0.08	**0.0678**	0.4	**0.0132**	1
SSS-Noself	0.08	**0.0630**	0.4	**0.0126**	1
SSS-Bound-0	0.08	**0.0400**	0.4	**0.0080**	1

Proposition 1: The activation probabilities of node j computed by the *Path Method* (π_j^p), by *SteadyStateSpread* (π_j^s), by *SSS-Bound-0* ($\pi_j^{t=0}$) and by *SSS-Noself* (π_j^n) satisfy:

$$\pi_j^{t=0} \leq \pi_j^p \leq \pi_j^n \leq \pi_j^s$$

The proof of this proposition is omitted for the sake of space.

V. EXPERIMENT

We study different networks to compare the results of Monte Carlo simulation, the *Path Method*, *SteadyStateSpread*, *SSS_Noself* and *SSS-Bound-t*.

A. Data Sets

There are two datasets for our experiments.

For the first dataset, we construct a series of bidirectional grid graphs with a parameter m such that the mth grid graph contains m^2 nodes. As shown in Fig. 4, initially we construct a circuit of 4 nodes when $m = 2$, then a grid graph of m^2 nodes is generated as m increases. It contains the grid graph of $(m-1)^2$ nodes as a subgraph and its nodes interact with each other. For each edge (i, j), we uniformly at random select $p_{i,j}$ from the set $\{0.1, 0.2, 0.5\}$. We represent this dataset as **Series-Grid**.

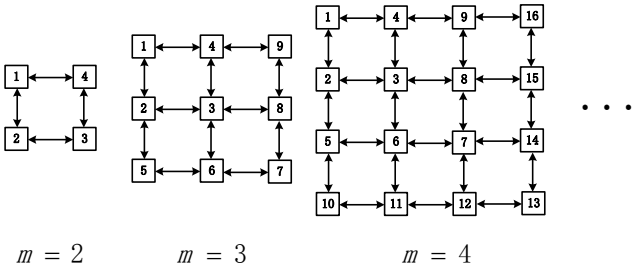


Fig. 4. Series of grid graphs

For the second part, we use a real-world dataset—**airportsinUS**[1]. It is a weighted network of the 500 airports with the largest amount of traffic from publicly available data in the United States. Nodes represent US airports and edges represent air travel connections among them. There are 5960 edges in total. Based on the weights $w_{i,j}$ of edges, we obtain $p_{i,j}$ by $w_{i,j} \setminus \sum_i w_{i,j}$.

All approaches are implemented in MATLAB. All experiments are run on a PC with 2.40GHz Intel Core i5 Processor and 8GB memory.

[1]https://sites.google.com/site/cxnets/usairtransportationnetwork

B. Experiment Results

Firstly, we present the computation of activation probabilities on Series-Grid using Monte-Carlo simulation, *Path Method*, *SteadyStateSpread* and *SSS-Noself*. We randomly select one node as seed node for the grid graphs with $m = \{2, 3\}$ and two nodes for the grid graphs with $m = \{4, 5, 6, 7\}$. The result of Monte Carlo simulation is the average of 10,000 times of simulations. We set $\varepsilon^* = 10^{-8}$ for the iterations of *SteadyStateSpread* algorithm and *SSS-Noself* algorithm. The sum of activation probabilities of nodes in Series-Grid with $m = \{2, 3, 4, 5, 6, 7\}$ using the four methods above is shown in Table III. The value for $m = \{4, 5, 6, 7\}$ by the *Path Method* is not given since the running time is more than 8 hours, i.e., out of time (o.o.t). As a particular case, we list activation probability of each node for the grid graph with $m = 3$ in Table IV to show the difference of every node by these four methods. We can observe that the result by *SSS-Noself* is always between the exact result by the *Path Method* and the result by *SteadyStateSpread* for both activation probability of each node and sum of activation probabilities of all node. It verifies the relationship among these three methods in Proposition 1. Moreover, it shows that our *SSS-Noself* algorithm provides more precise results than *SteadyStateSpread* algorithm.

TABLE III

SUM OF ACTIVATION PROBABILITIES BY MONTE CARLO SIMULATION, *Path Method*, *SteadyStateSpread*, AND *SSS-noself* ON SERIES-GRID WITH $m = \{2, 3, 4, 5, 6, 7\}$

Method	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$
Monte Carlo simulation	1.4343	1.9055	3.2171	4.5072	4.5527	5.1308
Path Method	1.4428	1.9098	o.o.t	o.o.t	o.o.t	o.o.t
SteadyStateSpread	1.4467	2.0936	5.2166	5.8790	7.0053	13.1089
SSS-Noself	1.4428	1.9502	4.0661	5.0710	5.7296	9.1891

TABLE IV

ACTIVATION PROBABILITIES BY MONTE CARLO SIMULATION, *Path Method*, *SteadyStateSpread*, AND *SSS-Noself* ON SERIES-GRID WITH $m = 3$

Node	1	2	3	4	5	6	7	8	9
Monte Carlo simulation	1	0.5076	0.0962	0.1099	0.1048	0.0302	0.0151	0.0168	0.0249
Path Method	1	0.5032	0.1009	0.1136	0.1049	0.0302	0.0161	0.0169	0.0238
SteadyStateSpread	1	0.5392	0.1349	0.1475	0.1317	0.0535	0.0292	0.0255	0.0320
SSS-Noself	1	0.5049	0.1119	0.1182	0.1093	0.0345	0.0206	0.0227	0.0280

Secondly, we compare the running time of these four methods for the activation probability computation, shown in Table V. The *Path Method* takes exponential time to give exact results as the size of network increases. *SSS-Noself* provides better results than *SteadyStateSpread* with an acceptable increase of computation time for the considered small networks.

The second part of our experiment is performed on airportsinUS network data. We evaluate the sum of activation probabilities for all nodes by *SteadyStateSpread*, *SSS-Noself* and *SSS-Bound-t* given different sizes of seed sets, shown in Fig. 5. The seven seed sets are randomly generate with

TABLE V

RUNNING TIME FOR ACTIVATION PROBABILITY COMPUTATION IN SERIES-GRID WITH $m = \{2, 3, 4, 5, 6, 7\}$

Running time (s)	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$
Monte Carlo simulation	0.48	1.25	0.95	1.40	1.82	1.58
Path Method	0.11	0.31	o.o.t	o.o.t	o.o.t	o.o.t
SteadyStateSpread	0.01	0.02	0.10	0.09	0.34	0.55
SSS-Noself	0.01	0.06	0.66	1.42	4.80	11.41

the size $|\phi_0| = \{1, 5, 10, 15, 20, 25, 30\}$. The tolerance is fixed as $\varepsilon^* = 0.01$ for *SteadyStateSpread* and *SSS-Noself*. The path bound t is chosen from $\{0, 1, 5, 10, 15, 20, 25, 30\}$ for *SSS-Bound-t*. We can observe that the sum of activation probabilities by *SteadyStateSpread* are no less than that by *SSS-Noself*. Since we have pointed $\pi_j^p \leq \pi_j^n \leq \pi_j^s$, $j \in \mathcal{V}$ in Proposition 1, although the results of *Path Method* are difficult to obtain for big networks, we can figure out that *SSS-Noself* is more precise than *SteadyStateSpread*. Moreover, we can discover that the results of *SSS-Bound-t* increase as the path bound t's increase. As a lower-bound of the exact solution, the results of *SSS-Bound-0* is the smallest among all algorithms.

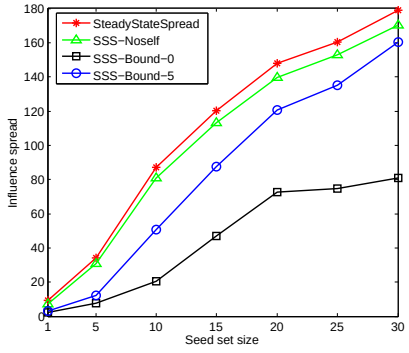


Fig. 5. Sum of activation probabilities by *SteadyStateSpread*, *SSS-Noself*, *SSS-Bound-0* and *SSS-Bound-5* on airportsinUS network data

However, as shown in Table VI, the running time of *SSS-Noself* is much longer than *SteadyStateSpread* when the size of the network is large since *SSS-Noself* needs one more pass of all nodes in a network than *SteadyStateSpread*. For this reason we think that it may be necessary to further improve the efficiency of *SSS-Noself*. Compared with the running time of *SSS-Bound-t* in Table VII, *SSS-Bound-t* is obviously faster than *SteadyStateSpread* when $t = \{0, 1, 5, 10, 15, 20\}$. As t increases, the running time of these two approaches will be similar.

TABLE VI

RUNNING TIME FOR ACTIVATION PROBABILITY COMPUTATION BY *SteadyStateSpread* AND *SSS-Noself* ON AIRPORTSINUS NETWORK DATA GIVEN DIFFERENT SIZES OF SEED SETS $|\phi_0|$

Running time (s)	$\lvert\phi_0\rvert=1$	$\lvert\phi_0\rvert=5$	$\lvert\phi_0\rvert=10$	$\lvert\phi_0\rvert=15$	$\lvert\phi_0\rvert=20$	$\lvert\phi_0\rvert=25$	$\lvert\phi_0\rvert=30$
SteadyStateSpread	5.05	3.20	1.35	1.18	1.10	0.94	0.82
SSS-Noself	1143.62	892.77	719.44	467.74	512.77	383.86	350.49

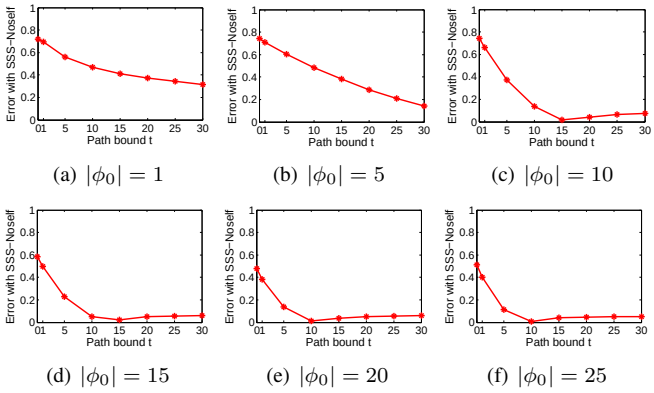
Fig. 6 shows the error between *SSS-Bound-t* and *SSS-Noself* which is measured by $|\sum_{j\in\mathcal{V}\setminus\phi_0}\pi_j^t - \sum_{j\in\mathcal{V}\setminus\phi_0}\pi_j^n|\setminus$

TABLE VII

RUNNING TIME FOR ACTIVATION PROBABILITY COMPUTATION BY SSS-BOUND-T ON AIRPORTSINUS NETWORK DATA GIVEN DIFFERENT SIZES OF SEED SETS $|\phi_0|$ WITH PATH BOUND $t = \{0, 1, 5, 10, 15, 20, 25, 30\}$

Running time (s)	$\|\phi_0\| = 1$	$\|\phi_0\| = 5$	$\|\phi_0\| = 10$	$\|\phi_0\| = 15$	$\|\phi_0\| = 20$	$\|\phi_0\| = 25$	$\|\phi_0\| = 30$
$t = 0$	0.12	0.12	0.11	0.14	0.12	0.11	0.12
$t = 1$	0.16	0.22	0.16	0.18	0.16	0.15	0.16
$t = 5$	0.26	0.28	0.28	0.30	0.27	0.27	0.28
$t = 10$	0.36	0.38	0.38	0.40	0.38	0.37	0.37
$t = 15$	0.61	0.57	0.56	0.58	0.56	0.56	0.57
$t = 20$	0.57	0.64	0.63	0.65	0.64	0.63	0.62
$t = 25$	0.87	0.87	0.86	0.88	0.86	0.86	0.86
$t = 30$	0.90	0.90	0.91	0.91	0.89	0.88	0.89

$\sum_{j \in \mathcal{V} \setminus \phi_0} \pi_j^n$. We do not present the curve for $|\phi_0| = 30$ due to the limit of space, but point out that it is similar to the curves for $|\phi_0| = \{15, 20, 25\}$. We can find that in the beginning the error decreases as the path bound t's increases. At certain path bound the error is the smallest and then increases a bit. It shows that the path bound corresponding to the smallest error varies with different seed set size.



(a) $|\phi_0| = 1$ (b) $|\phi_0| = 5$ (c) $|\phi_0| = 10$

(d) $|\phi_0| = 15$ (e) $|\phi_0| = 20$ (f) $|\phi_0| = 25$

Fig. 6. Error between *SSS-Bound-t* and *SSS-Noself* on airportsinUS network data given different sizes of seed sets $|\phi_0|$ with path bound $t = \{0, 1, 5, 10, 15, 20, 25, 30\}$

VI. CONCLUSIONS

In this paper, we focus on analyzing different approaches for the influence spread computation through a network. We initially propose an approach which can compute the influence spread exactly called the *Path Method*. We consider the elements resulting in the inaccuracy of *SteadyStateSpread*: the dependent relation between nodes and the existence of circuits. Furthermore, we show how to compute a lower approximation of activation probabilities by *SSS-Bound-t* and propose an improved algorithm called *SSS-noself* which partially decreases the error caused by circuits.

Aware of the factors which cause the inaccuracy of *SteadyStateSpread*, proposing new algorithms to improve the effectiveness will be the objective of our future work. Besides, since that the *SSS-noself* run for much time in large-scale networks, another interesting line of research could be to improve the efficiency of the *SSS-noself*. We believe that these computational approaches to determine influence diffusion can also be used to solve problems of influence maximization and minimization: our future research will explore these issues.

REFERENCES

[1] D. Rosa and A. Giua. On the spread of innovation in social networks. In *4th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 46(27): 322-327, 2013.

[2] D. Kempe, J.M. Kleinberg and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137-146, 2003.

[3] W. Chen, C. Wang, Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1029-1038, 2010.

[4] C. Zhou, P. Zhang, J. Guo, X. Zhu, and L. Guo. Ublf: An upper bound based approach to discover influential nodes in social networks. In *ICDM*, pp. 907-916, 2013.

[5] D. Kempe, J.M. Kleinberg and É. Tardos. Influential Nodes in a Diffusion Model for Social Networks. In *ICALP*, pp. 1127-1138, 2005.

[6] C.C. Aggarwal, A. Khan, X. Yan. On flow authority discovery in social networks. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 522-533, 2011.

[7] Y. Yang, E. Chen, Q. Liu, B. Xiang, T. Xu, and S. A. Shad. On approximation of real-world influence spread. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 548-564, 2012.

[8] E. Schechter. Handbook of analysis and its Foundations. *Academic Press*, 1996.

[9] P. Domingos, M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 57-66, 2001.

[10] B. Ryan, N.C. Gross. The diffusion of hybrid seed corn in two lowa communities. *Rural sociology*, 8(1): 15, 1943.

[11] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 199-208, 2009.

[12] J.S. Coleman, E. Katz, and H. Menzel. Medical innovation: A diffusion study. *Bobbs-Merrill Co*, 1966.

[13] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 420-429, 2007.

[14] S. Kakutani. A generalization of brouwer's fixed point theorem. *Duke Mathematical Journal*, pp. 457-459, 1941.

[15] J.C.P. Bus. Convergence of Newton-like methods for solving systems of nonlinear equations. *Numerische Mathematik*, 27(3): 271-281, 1976.

[16] C. Grosan and A. Abraham. Multiple solutions for a system of nonlinear equations. *International Journal of Innovative Computing, Information and Control*, 4(9): 2161-2170, 2008.

[17] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3): 211-223, 2001.

[18] M. Kimura, K. Saito. Tractable models for information diffusion in social networks. *Knowledge Discovery in Databases: PKDD*, pp. 259-271, 2006.