DOI: 10.15439/2014F261
ACSIS, Vol. 2

The inverse infection problem

András Bóta
University of Szeged,
Institute of Informatics,
P. O. Box 652.,
6701 Szeged, Hungary
Email: bandras@inf.u-szeged.hu

Miklós Krész
University of Szeged,
Gyula Juhász Faculty of Education,
Boldogasszony sgt. 6,
6720 Szeged, Hungary
Email: kresz@jgypk.u-szeged.hu

András Pluhár
University of Szeged,
Institute of Informatics,
P. O. Box 652.,
6701 Szeged, Hungary
Email: pluhar@inf.u-szeged.hu

***Abstract*—The applications of infection models like the Linear Threshold or the Domingos-Richardson model requires a graph weighted with infection probabilities. In many real-life applications these probabilities are unknown; therefore a systematic method for the estimation of these probabilities is required. One of the methods proposed to solve this problem, the Inverse Infection Model, was originally formulated for estimating credit default in banking applications. In this paper we are going to test the capabilities of the Inverse Infection Model in a more controlled environment. We are going to use artificially created graphs to evaluate the speed and the accuracy of estimations. We are also going to examine how approximations and heuristics can be used to improve the speed of the calculations. Finally, we will experiment with the amount of a priori information available in the model and evaluate how well this method performs if only partial information is available.**

I. Introduction

THE STUDY of infection processes has roots in two seemingly different fields of research: sociology and the medical sciences. In the latter, it was used to model the spread of epidemics [9]. Applications focused on prevention, and the identification of the “choke points" during an epidemic. In the former, the spreading of information or opinions came into focus. One of the earliest models in sociometry, Granovetter's Linear Threshold [12] model is still considered to be a viable description of information diffusion.

In economics, Domingos and Richardson developed the Independent Cascade model (IC) [10] for the purpose of viral marketing. They proposed the influence maximization problem, that is to find the set of k initial infectors for any k that results in the largest expected infection. Kempe et al. [15], [16] proved the influence maximization problem was NP-hard, proposed a greedy algorithm for it, and also showed that the generalization of the IC model is in fact an equivalent of the Linear Threshold model. They also used random simulations to approximate the vertex infection probabilities, and they choose an arbitrary constant for edge infection probabilities. This result stresses the importance of the exact computation of vertex infection probabilities. This problem was proven to be #P-complete by Cao [4].

Computing the maximal infection or the exact probabilities of infection with any kind of model requires a weighted network, that is the edge infection probabilities must be available. This information is usually not known beforehand. In most real-life applications, the edges are considered to be some constant, or estimated using intuition guided trial-and-error method based on known edge or vertex attributes. Recently, a few papers were published in this topic discussing systematic approaches for the estimation of edge infection probabilities. In some of them [11], [17], the steps or iterations of the infection process are assumed to be known, which is realistic in the case of twitter or blog-based networks.

The Inverse Infection Problem, an application-driven approach was proposed recently by the authors [1] for the prediction of credit default in bank transaction networks. Unlike the above mentioned methods, this does not require information on the individual steps of the infection process. Instead it builds on other available data, such as estimations of the probabilities of default for individual companies and additional information characterizing the connection between the companies.

Based on the good results of this method in applications, our goal in this paper is to provide a solid foundation to the Inverse Infection Problem in a more controlled environment. Our method is based on the Generalized Cascade (GC) model [3], a generalization of the Independent Cascade model. To compute the edge infection probabilities themselves we will use a meta-heuristic: the Particle Swarm Algorithm of Kennedy and Mendes [14].

The paper is constructed as follows. In the next section we will give a short introduction into infection mechanisms, define the GC model, and the Inverse Infection Problem. In Section III we will describe several ways to accurately estimate the infection probabilities, including gradient-based methods and Particle Swarm Optimization. Then we will discuss various options to customize the estimations including heuristics of the GC model, choices for attribute functions and the number of patterns required to accurately estimate the infection probabilities.

II. Problem definition

The process of infection takes place on a graph G, where $V(G)$ denotes the set of vertices, and $E(G)$ denotes the set of edges. While most traditional models require directed edges, depending on the application, they can be easily modified to handle undirected ones. We also need to know the edge infection probabilities, that is a weight $w_e \in [0,1]$ for each edge e.

The notion of states is important. Each vertex of the network has a state of infection. The number of states and the transitions between them are governed by the specific model. One of the most basic approaches, the SIR model, [9] has three states: Susceptible, Infected and Recovered. Infected nodes infect susceptible ones, but after a certain period, which is usually a parameter of the model, they may recover, no longer infectious. Models in epidemics have a variety of states and the transitions between them are often more complicated. Models in economics or models describing information diffusion can be considered simpler. In the case of the Independent Cascade model, there are three states loosely corresponding to the ones in the SIR model and the infection period only lasts for one iteration. These three states are: susceptible, just infected (and still infectious), infected (but no longer infectious).

Most infection processes are also iterative, that is the process takes place in discrete time steps. Those models, that allow nodes to become susceptible again some time after becoming infected, may not terminate. It is easy to see, that the IC model terminates in finite steps.

A. Infection Models

Any infection model can be described as a process, that has two inputs: the first one is a weighted graph, where the edge weights are probabilities. The second input is the set of initial infectors $A_0 \subset V(G)$. These nodes are considered as infected at the beginning of the process. The process terminates at iteration t, and results in the set of infected nodes $A = \bigcup_{i=0}^{t} A_i$.

The specific way one vertex infects another varies depending on the model. In the case of the IC model [10], let $A_i \subseteq V(G)$ be the set of nodes newly activated in iteration i. In the next iteration $i+1$, each node $u \in A_i$ tries to activate its inactive neighbors $v \in V \setminus \cup_{0 \le j \le i} A_j$ according to the edge infection probability $w_{u,v}$, and v becomes active in iteration $i+1$, if the attempt is successful. If more than one node is trying to activate v in the same iteration, the attempts are made independently of each other in an arbitrary order within iteration $i+1$. If $A_t = \emptyset$, the process terminates in iteration t. It is easy to see, that the process always terminates in a finite number of iterations.

B. Generalized Cascade Model

Following the works of Bóta et al. [3], we can generalize this model in the following way. Instead of using vertex sets for representing the initial infectors, we work with two probability distributions. The *a priori* distribution defines the probability, that a vertex becomes infected on its own, independently of other vertices at the beginning of the process. The *a posteriori* defines the probability, that a vertex becomes infected at the end of the process. For all vertices $v \in V(G)$, we will denote the a priori probability of infection as p_v, the a posteriori as p'_v.

In some applications, an estimate of one or both of the above described probability distributions is available. For example, in the case of the banking application [1], [7] an accurate estimation of the probability of default for each company was given by standard models used by the bank[1]. Another application in telecommunications uses estimations for the probability of churn using similar methods. If such estimations are not available we can resort to a crude but effective method. Suppose we can observe the beginning and the end of the infection process k times. By counting the frequencies of infection, for all vertices v, how many times did v belong to A_0 or A we can construct the respective probability distributions. The accuracy of the estimation obviously depends on k, but k does not have to be a large number. We will show in section IV.D, that 6-8 observations are enough to produce outputs with acceptable quality.

Based on these remarks and formulations, we can define the Generalized Cascade model [2]:

The Generalized Cascade Model: *Given an appropriately weighted graph G and the a priori infection distribution p_v, the model computes the a posteriori distribution p'_v for all $v \in V(G)$.*

The infection process itself is the IC model, although other models might also be used for different applications. We have chosen the Independent Cascade model as the basis of our method, because it performs well in modeling infection-like processes in business applications [7]. Alternatively, this model can be considered as a general framework of infection.

Unfortunately, the computation of the a posteriori distribution in the IC model is #P-complete. There are several existing heuristics to provide estimations of p'_v [5], [6], including the ones the authors proposed in [2]. Two of these are Monte Carlo based simulations. *Complete Simulation* is a direct adaptation of the idea of Kempe et al. to the framework of the GC model. The basis of the idea is the notion of reachability. By selecting the edges $(u,v) \in E(G)$ independently of each other according to their infection probabilities $w_{u,v}$, they construct an unweighted graph which is a realization of the infection process. Any vertex, that can be reached from any initially infector is considered to be infected. We can adapt this process into the GC model by computing a large number of individual runs of the model and counting the frequencies of infections (both a priori and a posteriori). The process has an unfortunate property: the frequency (or sample size) must be high enough to reduce the standard deviation characteristic of Monte Carlo based methods.

The *Edge Simulation* method decreases the standard deviation of the previous method. In each run, a subgraph containing all of the vertices able to infect the individual vertex v is constructed for all vertices $v \in V(G)$. This way the a posteriori infection of v can be computed directly in each run. The results of individual runs are averaged. The authors have proposed two additional heuristics: In the *Neighborhood Bound Heuristic* a tree is constructed from the 2-neighborhood

[1]The BASEL II default probabilities were computed using vertex attributes.

of a given vertex v representing all possible routes of infection. Both the tree and the a posteriori infection of v can be computed in a short time, resulting in a very fast heuristic. The *Aggregated Linear Effect* model is a linear approximation of the mechanism of the IC model. A more detailed description of these methods can be found in [2].

C. Inverse Infection Problem

Based on the framework of the Generalized Cascade model, we can define the Inverse Infection Problem.

Inverse Infection Problem: *Given an unweighted graph G, the a priori and the a posteriori probability distributions p_v and p'_v, compute the edge infection probabilities w_e for all $e \in E(G)$.*

Directly estimating each individual edge in a graph is computationally infeasible even in small graphs. However, in real-life applications the probability of infection between vertices is a combination of other properties of the edges, vertices and the graph itself. We are going to take advantage of this fact to simplify computations and make the problem solvable in reasonable time. We are going to assume, that on each edge there are several *attributes*[2], and the infection probability of the edge is a parametrized*function* of these attributes[3]. This way, only the *coefficients* of this function have to be estimated, which is a small number compared to the number of edges.

There are multiple ways to define these functions. In this paper, we are going to consider, that there is a polynomial function f_i on each individual attribute $a_i, i = 1, \ldots, \ell$, where ℓ is the number of attributes. Then, a normalized sum or product is calculated from each $f_i(a_i)$ resulting in the infection probability w_e. The degree of these polynomials should be low, but we allow different polynomials on different edges, with possibly different degrees. If the maximum degree of these polynomials is f_{max}, then there are at most $(f_{max} + 1)\ell$ coefficients to estimate.

III. Estimation with learning methods

To provide a solution for the Inverse Infection Problem, we have developed the following learning algorithm. The problem definition states, that the a posteriori distribution is required as an input of any algorithm. In the case of a learning algorithm, it is considered as a test or reference dataset. By taking the a priori distribution we compute an estimation of the a posteriori distribution with some initially random starting coefficients. Then, an error function calculates the difference between the reference set and the newly calculated infection values. Our goal is finding the global minimum of this error function: the difference between the a posteriori vertex infections. Using an optimization algorithms, we can efficiently estimate the coefficients of the attribute-functions and thus the edge weights.

[2]Vertex attributes can be easily converted into edge attributes.

[3]It is possible, that some of these attributes have no influence on the infection probability, but we expect the method to ignore the effect of these.

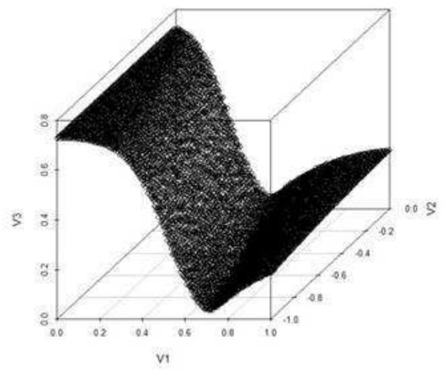

Fig. 1. The error surface of an IIP with one edge attribute and $f_1(a_1) = c_0 * a_1 + c_1$ as the attribute function. Root mean squared error was used for the evaluation.

A. Previous approaches and experiences

We have tried several optimization algorithms, including more simple ones, like grid search, gradient-based methods and meta-heuristics. Our first analysis was performed on banking data where we used grid search for optimization. While this early approach provided promising results [7], it was clear that further refinement of the optimization algorithm was required. Later, we have implemented a multi agent gradient based method, and compared the performance of it with our previous results [3]. The gradient method provided more accurate estimations, but highlighted several unfortunate properties of the problem itself.

Our first observation was that the error function was noisy. This comes from using Monte Carlo methods to approximate the IC model, since the deviation of these simulations makes different runs with the same coefficient values have different results. The noise can be reduced by increasing the frequency parameter of the simulation, but this also increases the time complexity of the method [3].

The second observation was that the problem is underdetermined. Different edge weight configurations can result in the same infection pattern, the same a posteriori distribution. This results in alleys and plateaus on the error surface. In the case of the example on Figure 1, the global minimum is in the middle of the alley. Even in this simple example neither algorithms are able to reliably find the best solution.

Grid search had serious performance issues both in finding the global minimum and in time complexity. Due to its search pattern, its precision is simply not enough to tackle with this surface, and it also scales exponentially with the number of coefficients. The gradient method also performs poorly: it easily gets lost on the alleys and plateaus especially if they are noisy as well. As a consequence, it rarely finds a solution close to the global minimum, and the number of steps it takes to find a solution at all can be quite high.

We have tried several error functions, mainly vector distance measurements, and ROC evaluation. One of our first experiences was, that the latter is not enough to properly guide the optimization method to the global minimum, so we have shifted our attention to other measurements, and finally settled on the root mean squared error. In this work we are going to use the RMSE as an error measurement, that is we are looking for the minimum of

$$\sqrt{\frac{1}{|V(G)|}\sum_{v\in V(G)}(\vec{\tilde{p}}'_v - \vec{p}'_v)^2}, \tag{1}$$

where $\vec{\tilde{p}}'_v$ denotes the estimated a posteriori infection of vertex v.

B. Particle-Swarm Optimization

In order to handle the above mentioned problems, we have decided to implement the particle swarm optimization algorithm of Kennedy [13]. This is an iterative method based on the interaction of multiple agents or "particles". Each agent corresponds to a different coefficient configuration, representing a coordinate in the parameter space of the problem[4].

Apart from the coordinates themselves, the agents also have a velocity. In each iteration the position of an agent is updated by adding its velocity. The velocity of the agent is computed using the best solution the agent has found and the best solutions of the neighboring agents; the goodness of the solution is measured by evaluating the error function on the coordinates visited by the particles. Agents are connected to each other according some topology describing the neighborhood of each agent.

The specific way the velocities of the agents are updated and the topology itself is not fixed: there are various approaches in the literature for specific applications and for more general problem solving. In our work we have followed the recommendations of Kennedy and Mendes [14], and found, that it performs well in finding coefficient configurations close to the global minimum.

We have used the Fully Informed Particle Swarm published in [14] with 9 agents in a von Neumann neighborhood[5]. The position and the velocity of the agents are updated according to the following equations:

$$\vec{v}_i \leftarrow \chi\left(\vec{v}_i + \sum_{n=1}^{N_i}\frac{U(0,\varphi)(\vec{b}_{nbr(n)} - \vec{x}_i)}{N_i}\right), \tag{2}$$

$$\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i, \tag{3}$$

where $\vec{x}_i$ and $\vec{v}_i$ denotes the coordinate and velocity of particle i, $U(min, max)$ is a uniform random number generator, $\vec{b}_i$ is the best location found so far by particle i, N_i is the number of neighbors i has and $nbr(n)$ is the nth neighbor of i. The formula has two parameters: χ is the constriction coefficient

[4]Again, the subject of the optimization is the coefficient values of the attribute function(s)

[5]Each agent has four neighbors in a grid, connected to the upper, lower, left and right, while wrapping around the edges.

Algorithm 1 Particle Swarm Optimization

```
 1: for all $a_i$ do
 2:     Initialize $\vec{x}_i$ for agent $a_i$ within the boundaries of the
        search space
 3:     Initialize $\vec{v}_i$ for agent $a_i$
 4:     Set $\vec{b}_i \leftarrow \vec{x}_i$
 5:     Select the neighbors of $a_i$ according to the topology
 6: end for
 7: repeat
 8:     for all $a_i$ do
 9:         Update $\vec{v}_i$ according to equation 2
10:         Update $\vec{x}_i$ according to equation 3
11:         Calculate the error function $e(\vec{x}_i)$ in position $\vec{x}_i$
12:         if $e(\vec{x}_i) < e(\vec{b}_i)$ then
13:             $\vec{b}_i \leftarrow \vec{x}_i$
14:         end if
15:     end for
16: until termination criterium is met
```

and φ is the acceleration constant. Again, we have used the recommendations of Kennedy et al., and set $\chi = 0.7298$ and $\varphi = 4.1$.

At the beginning of the search, the agents are initialized with zero velocities and random starting coordinates within some reasonable bounds of them. Then in each iteration these two vectors are updated according to equations 2 and 3 in a synchronized manner. The search is completed if the global minimum found considering all agents does not change for five consecutive iterations. We have experimented with other values and found, that increasing it does not improve the quality of the results, and decreasing it does not reduce the running time considerably.

IV. Evaluation

The most natural way to evaluate the stability of the optimization method is by counting the average and maximum number of iterations the method takes before it finishes. However, the quality of the solution of the Inverse Infection Problem depends on additional factors; we will discuss three of these. The first one is the choice of the attribute functions. Choosing an appropriate function is important, since depending on the available attributes this function either maps into the $[0, 1]$ interval directly or some additional form of normalization is required. The second one is the choice of heuristics applied for the GC model. These have a serious impact on both the accuracy and the running time of the learning method. The third factor is the number of learning patterns available. In case the exact a priori and a posteriori infection probabilities are not available, the only thing to do is to rely on counting the frequencies of infections. In real life we cannot hope to witness an infection process on any network in more than a handful of times. It is therefore necessary to investigate the sensitivity of our method to low-quality inputs.

As a basis of our analysis we have used graphs generated with the forest fire method of Leskovec et al. [18]. We have

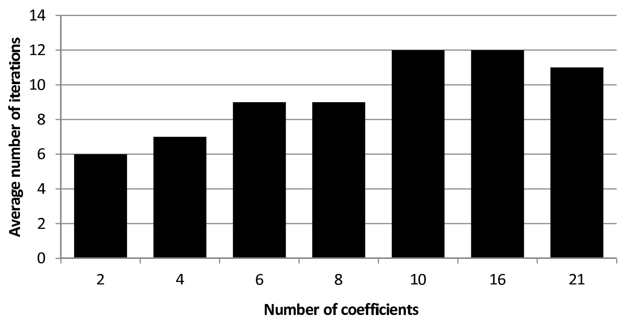

Fig. 2. The average number of iterations with different function configurations.

created a series of graphs of sizes $n = 1000, \ldots, 100000$, with parameters $p = 0.37$ and $p_b = 0.32$, for forward and backward burning probabilities, respectively. We have assigned a number of edge attributes $a_i, i = 2, \ldots, 10$ to these networks. These attributes are randomly generated: they were drawn independently from a uniform distribution between $[0, 0.5]$. We have also used randomly generated a priori infections. The expected size of the set of initial infectors is $0.3{*}n$. If v is selected, then an a priori infection probability was drawn from an uniform distribution between $[0, 0.5]$, otherwise $p_v = 0$. We have used various attribute functions, description of these will be given in section IV.B. Finally for each network and each attribute function we have created an a posteriori infection distribution as the reference dataset. For this purpose we have used Compete Simulation with sample size $k = 10000$, because this method gives the best approximation of the original IC model.

A. Stability of the optimization

The performance of the optimization method itself can be measured in two ways. The distance between the solution found by the method and the global minimum is conveniently measured by the error function itself. However, the precision of the algorithm also depends on the heuristics used to approximate the Generalized Cascade model. Consequently, we will discuss this in the following sections.

The time complexity of the method is the sum of two distinct parts of the algorithm: evaluating points on the error surface and the search method itself. The latter consists of the repeated evaluation of the formula above, after initializing the neighborhood and the starting coordinates. Since the number of agents is small, this part of the algorithm is very fast, and has negligible impact on the running time of the overall method.

In each iteration every agent evaluates the error function. This evaluation is the computation of the GC model using the coordinates - coefficients of the given agent. The time complexity of this step heavily depends on the used heuristic. Altogether, we can say, that the time complexity of a single run is $s * a * h$, where s is the number of iterations, a is the number of agents (a constant) and h is the time complexity of the infection heuristic. This also means, that we can describe the time complexity of the algorithm by measuring the average or maximum number of iterations and multiplying it with the time complexity of the infection method and the number of agents. Breaking the time complexity of the method into two different factors makes sense because of another reason: the individual runs of the GC heuristics may be run on multiple threads simultaneously, significantly improving the speed of the method.

On Figure 2 we can see the average number of iterations for different numbers of coefficients. We have used a small network with $|V(G)| = 1000$. The point of interest here is, starting from a simple problem with only two coefficients to more complex ones, the expected number of iterations grows slowly, and stabilizes around 12. The maximum number of iterations remains bounded as well, even in the experiment with 21 coefficients, it does not go beyond 30. The results shown on Figure 2 were computed with 9 agents. We have tried this problem with 16 agents as well and got similar results. If we compare the different infection heuristics, they perform similarly, with the non-Monte Carlo methods finishing slightly sooner, usually by 4-5 iterations.

We can conclude, that the Particle-Swarm Optimization method described in this section is able to solve the Inverse Infection Problem with satisfying results. The algorithm is very stable, and even in the worst case, it finishes within 30 iterations. We will evaluate the precision and running times of this method considering different heuristics of the Generalized Cascade model in section IV.C. We will also discuss choices for attribute functions, and the number of patterns required to get good estimations of the edge infection probabilities.

B. Choice of attribute functions

We have seen, that in our model, the edge infection probabilities are computed from some additional information on the edges in the form of edge attributes by so-called attribute functions. The choice of these attribute functions is an important part of our method. A natural requirement of this choice is, that it must result in infection probabilities: it must map into the $[0, 1]$ interval.

There are two approaches to this problem: the first one is to construct problem-specific functions, taking into account the structure of the network, the nature of the infection model and the number and domain of the attributes. This way it is possible to calculate the infection probabilities directly, without any form of additional normalization. This is the obvious choice if the above mentioned information is available.

If we do not have this information, we can try a more user-friendly approach. We can apply functions to the individual attributes, summarize them and finally normalize them. A variety of functions might be considered for this purpose. In our work, we have used low-degree polynomials for the individual attributes and simple addition or multiplication to join them. We have normalized the resulting edge infection probabilities according to

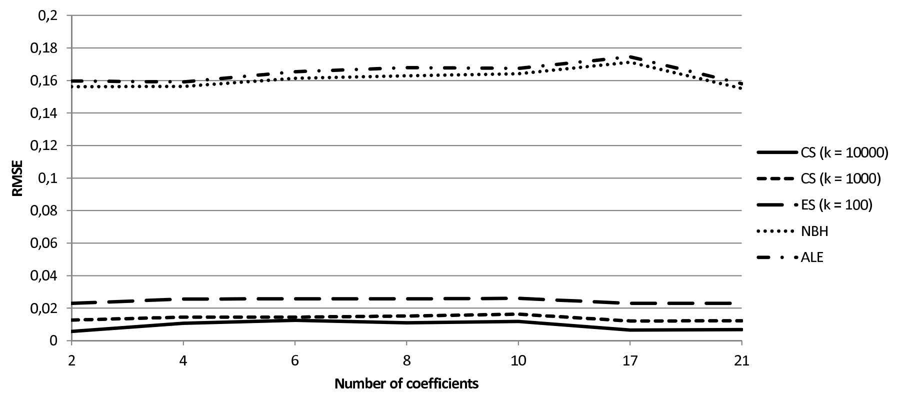

Fig. 3. The RMSE with different function configurations on a small network with $n = 1000$.

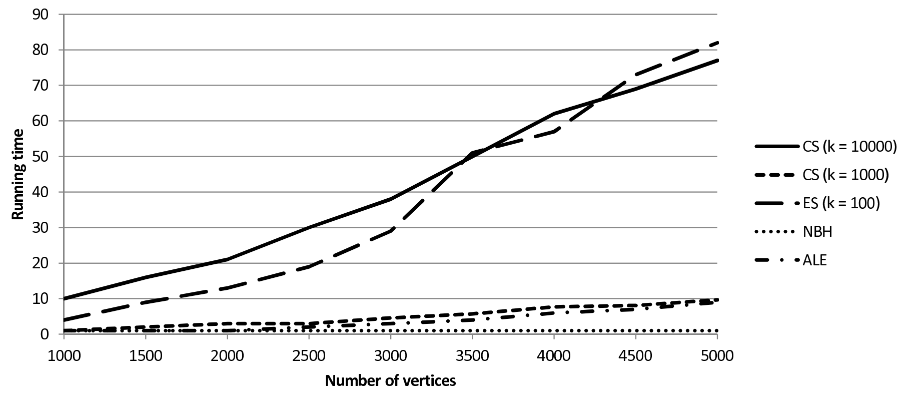

Fig. 4. The running time of the infection heuristics with different network sizes measured in seconds. Figure used with the permission of the authors [2].

$$\text{norm}(\vec{e}) = \frac{\vec{e} - \min(\vec{e})}{3(\max(\vec{e}) - \min(\vec{e}))}, \quad (4)$$

where $\vec{e}$ is a vector containing the infection probabilities for each individual edge. The reason why we have used the multiplier 3 in the denominator is, that according to our findings in the prediction of default events on banking data, the edge infection probabilities are low [7]. The normalizer function obviously distorts the shape of the individual attribute functions, but in real-life problems a simple weighted, normalized sum of attributes is sufficient to produce acceptable results.

In this paper, we have used seven attribute function configurations, a_i denotes attribute i and c_j denotes coefficient j:

- Weighted sum of two attributes: $c_1 a_1 + c_2 a_2$, two coefficients in total.
- Weighted sum of four attributes: $\sum_i c_i a_i, i = 1, 2, 3, 4$, four coefficients in total.
- Weighted sum of six attributes: $\sum_i c_i a_i, i = 1, \dots, 6$, six coefficients in total.
- Weighted sum of eight attributes: $\sum_i c_i a_i, i = 1, \dots, 8$, eight coefficients in total.
- Weighted sum of ten attributes: $\sum_i c_i a_i, i = 1, \dots, 10$, ten coefficients in total.
- Sum of quadratic polynomials with eight attributes $c_1 + \sum_i (c_{2i} a_i^2 + c_{2i+1} a_i)$, $i = 1, \dots, 8$, 17 coefficients in total.
- Sum of quadratic polynomials with ten attributes $c_1 + \sum_i (c_{2i} a_i^2 + c_{2i+1} a_i)$, $i = 1, \dots, 10$, 21 coefficients in total.

In section IV, we have tested the effect of these function configurations on the stability and accuracy of the optimization method. Details of these can be found in the appropriate subsections.

C. Accuracy and the choice of heuristics

Previously, in section II.B, we have given short descriptions of some heuristics of the GC model [2]. In this section we will evaluate the performance of them in relation with the learning method described above. Complete Simulation is a direct adaptation of the idea of Kempe et al. [15], it can be considered as the best approximation of the original IC model. Therefore, we will use CS with sample size $k = 10000$ to create an a posteriori distribution as a reference set. Then, we

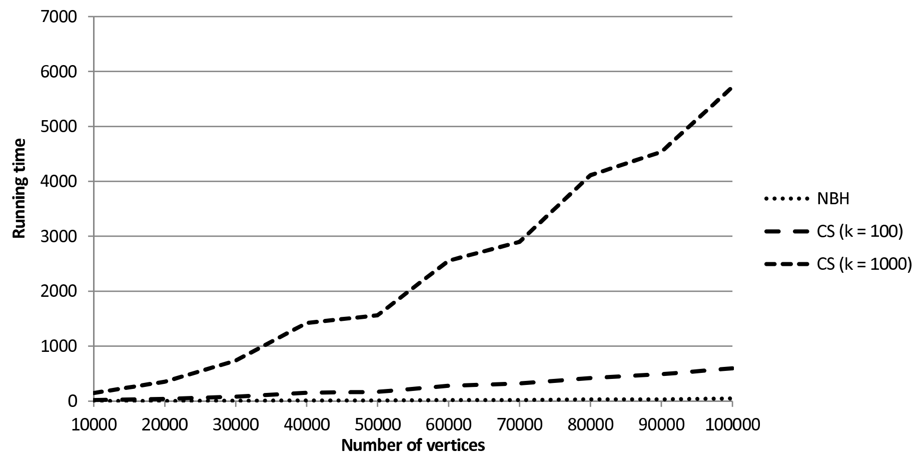

Fig. 5. The running time of the infection heuristics on large networks measured in seconds.

will use each heuristic together with the learning method to compute the edge infection probabilities:

- Complete Simulation (CS) with sample size $k = 10000$, a very accurate simulation.
- Complete Simulation (CS) with sample size $k = 1000$, a very fast simulation.
- Edge Simulation (ES) with sample size $k = 100$, a simulation based heuristic.
- Neighborhood Bound Heuristic (NBH), an extremely fast lower approximation.
- Aggregated Linear Effect (ALE) model, a de Groot [8] based simplification of the infection process.

Since the running time and the accuracy of the heuristics are different, we are going to use two different datasets to evaluate their performance. First, small networks with $|V(G)| \leq 5000$ will be used to make general observations, then we will test the more robust heuristics on large networks with $|V(G)| = 10000, \ldots, 100000$. Our largest network has 100000 vertices and 2.3 million edges.

As we can see on Figure 3, the Monte Carlo based simulations of the GC model (CS and ES) are able to estimate the reference distribution well, with the measured error between $0.01 - 0.03$. The other two heuristics (NBH and ALE) are tailored to small edge infection probabilities with rare infections, hence they do not perform so well on this dataset. Note, that in some cases even an error of this magnitude is acceptable, and the time complexity of these methods allows them to handle larger networks. If we compare the results computed by using different attribute functions, we can see that they have minimal effect on the accuracy of the methods.

Our results on the running times[6] of these heuristics on small networks correspond with our previous findings [2]. The speed of the simulations are governed by the sample size. Complete Simulation is considerably faster than ES[7] because the latter focuses on the fast computation of smaller infections. By decreasing the sample size CS can tackle larger networks as well. The Neighborhood Bound Heuristic is able to compute the a posteriori infections of the largest networks within a minute, enabling our method to scale upwards and handle real-life datasets and networks with possibly millions of nodes and edges.

We can conclude, that in general, the use of Complete Simulation is recommended. Both its precision and accuracy are good on large graphs. If the infection probabilities are lower than our current dataset, the use of Edge Simulation is also advisable. The Neighborhood Bound Heuristic and the Aggregated Linear Effect model are able to handle even larger networks, yet this comes at the cost of a significantly lower precision.

D. *Number of patterns*

In many real-life applications the a priori or a posteriori probabilities of infections are unavailable. In this section we are going to assume, that the initial infection probabilities are given, but we only have a small number of observations on the a posteriori infections. We are going to simulate this on a small network by generating an a posteriori distribution using CS with $k = 1, \ldots, 10$, corresponding to $1, \ldots, 10$ observations.

We can see, that the proposed method gives a rough estimate of the vertex infection probabilities in only a few iterations. If we consider a threshold of 0.15 as an acceptable estimation, our method only requires 6 observations to reach it. However, it is important to keep in mind, that the method tries to create a posteriori infections close to the reference. The problem is underdetermined even with exact possibilities of vertex infection, with only a handful number of observations many attribute function configurations (and edge weights) may result in the same infection. The results in this section only imply, that our method is able to give one of these.

Different infection heuristics are shown on Figure 6, one can see, that the simulations have identical performance regardless

[6]We have implemented the methods in JAVA, and we have used a computer with an Intel i7-2630QM processor, and 8 gigabytes of memory.

[7]Note the sample sizes.

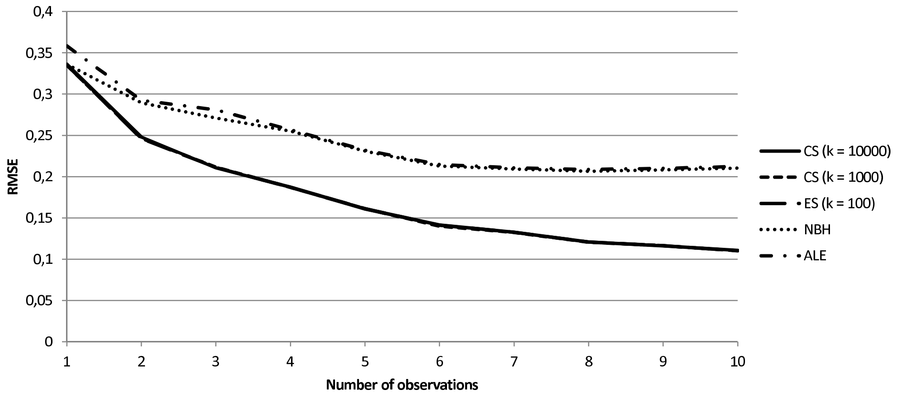

Fig. 6. The precision of the infection heuristics with a limited number of observations of the reference distribution.

of the sample size. As before, the non-Monte Carlo based methods perform poorly, their use is not recommended with low-quality inputs.

V. Conclusions

Our goal on this paper was to extend the previous results of the Inverse Infection Problem and its solution. We have given a detailed description and analysis of the Generalized Cascade Model, the Inverse Infection Problem and a Particle-Swarm Optimization algorithm capable of giving a good estimation of the latter. Several aspects of the method were investigated: We have tested the stability and accuracy of the optimization method, we have given a general approach to choose the correct attribute functions, we have examined the implications of choosing between the heuristics of the GC model and we have tested our method in low-quality inputs as well.

The given method is able to accurately predict the edge infection probabilities in a small number of iterations while the number of attributes and the shape of the attribute functions have only a small effect on this. Our method also handles low-quality inputs well. Of the infection heuristics we recommend the use of Complete Simulation, because it gives accurate results with acceptable standard deviation in reasonable time.

In our previous paper we have also given an application of this method in the prediction of credit default [7]. Our method was able to predict the default of the worst 5% of clients accurately.

Acknowledgment

The first and second authors were supported by the European Union and co-funded by the European Social Fund. Project title: “Telemedicine-focused research activities on the field of Mathematics, Informatics and Medical sciences. Project number”: TÁMOP-4.2.2.A-11/1/KONV-2012-0073

The third author was supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TÁMOP-4.2.2.C-11/1/KONV-2012-0013).

References

[1] A. Bóta, M. Krész and A. Pluhár, Applications of the Inverse Infection Problem on bank transaction networks. Submitted.

[2] A. Bóta, M. Krész and A. Pluhár, Approximations of the Generalized Cascade Model. *Acta Cybernetica* **21** (2013) 37–51.

[3] A. Bóta, M. Krész and A. Pluhár, Systematic learning of edge probabilities in the Domingos-Richardson model. *Int. J. Complex Systems in Science* Volume **1(2)** (2011) 115–118.

[4] Tianyu Cao, Xindong Wu, Tony Xiaohua Hu and Song Wang, Active Learning of Model Parameters for Influence Maximization. *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, eds. Gunopulos et al., Springer Berlin/Heidelberg, (2011) 280–295, http://dx.doi.org/10.1007/978-3-642-23780-5_28.

[5] Wei Chen, Yifei Yuan and Li Zhang, Scalable Influence Maximization in Social Networks under the Linear Threshold Model. *Proceeding ICDM ’10 Proceedings of the 2010 IEEE International Conference on Data Mining*, IEEE Computer Society (2010) 88–97, http://dx.doi.org/10.1109/ICDM.2010.118.

[6] Wei Chen, Chi Wang and Yajun Wang, Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM (2010) 1029–1038, http://doi.acm.org/10.1145/1835804.1835934.

[7] A. Csernenszky, Gy. Kovács, M. Krész, A. Pluhár, T. Tóth, The use of infection models in accounting and crediting. *Challenges for Analysis of the Economy, the Businesses, and Social Progress* Szeged (2009) pp. 617–623.

[8] M. H. DeGroot Reaching a Consensus. *Journal of the American Statistical Association*, **69** (345): 118–21, http://www.tandfonline.com/doi/pdf/10.1080/01621459.1974.10480137.

[9] O. Diekmann, J. A. P. Heesterbeek, Mathematical epidemiology of infectious diseases. Model Building, Analysis and Interpretation. *John Wiley & Sons,* 2000.

[10] P. Domingos, M. Richardson, Mining the Network Value of Costumers. *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, ACM (2001) 57–66, http://doi.acm.org/10.1145/502512.502525.

[11] A. Goyal, F. Bonchi, L.V.S. Lakshmanan Learning influence probabilities in social networks. *Proceedings of the third ACM International Conference on Web search and data mining.* ACM (2010) 241–250, http://doi.acm.org/10.1145/1718487.1718518.

[12] M. Granovetter, Threshold models of collective behavior. *American Journal of Sociology* **83**(6) (1978) 1420–1443, http://psycnet.apa.org/doi/10.1086/226707.

[13] J. Kennedy Particle Swarm Optimization. *Encyclopedia of Machine Learning*, Springer US (2010), 760–766, http://dx.doi.org/10.1007/978-0-387-30164-8_630.

[14] J. Kennedy, R. Mendes Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews.* **36** (4) (2006) 515–519, http://dx.doi.org/10.1109/TSMCC.2006.875410.

[15] D. Kempe, J. Kleinberg, E. Tardos, Maximizing the Spread of Influence though a Social Network. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM (2003) 137–146, http://doi.acm.org/10.1145/956750.956769.

[16] D. Kempe, J. Kleinberg, E. Tardos, Influential Nodes in a Diffusion Model for Social Networks. *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, Springer-Verlag (2005) 1127–1138, http://dx.doi.org/10.1007/11523468_91.

[17] M. Kimura, K. Saito, Tractable models for information diffusion in social networks. *Knowledge Discovery in Databases*, Lecture Notes in Computer Science Springer Berlin / Heidelberg, (2006), 259–271, http://dx.doi.org/10.1007/11871637_27.

[18] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations. *Proceedings of the1th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM (2005) 177–187, http://doi.acm.org/10.1145/1081870.1081893.