Estimation of Edge Infection Probabilities in the Inverse Infection Problem

András Bóta, Miklós Krész and András Pluhár

Abstract Several methods have been proposed recently to estimate the edge infection probabilities in infection or diffusion models. In this paper we will use the framework of the Generalized Cascade Model to define the Inverse Infection Problem—the problem of calculating these probabilities. We are going to show that the problem can be reduced to an optimization task and we will give a particle swarm based method as a solution. We will show, that direct estimation of the separate edge infection values is possible, although only on small graphs with a few thousand edges. To reduce the dimensionality of the task, the edge infection values can be considered as functions of known attributes on the vertices or edges of the graph, this way only the unknown coefficients of these functions have to be estimated. We are going to evaluate our method on artificially created infection scenarios. Our main points of interest are the accuracy and stability of the estimation.

1 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 [10]. 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 [13] model is still considered to be a viable description of information diffusion.

A. Bóta (✉) · A. Pluhár
Institute of Informatics, University of Szeged, P. O. Box 652, Szeged 6701, Hungary
e-mail: bandras@inf.u-szeged.hu

A. Pluhár
e-mail: pluhar@inf.u-szeged.hu

M. Krész
Gyula Juhász Faculty of Education, University of Szeged, Boldogasszony Sgt. 6,
Szeged 6701, Hungary
e-mail: kresz@jgypk.u-szeged.hu

© Springer International Publishing Switzerland 2016

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 610, DOI 10.1007/978-3-319-21133-6_2

In economics, Domingos and Richardson developed the Independent Cascade model (IC) [11] 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. [16, 17] 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 et al. [6].

Computing maximal infections 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 methods 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 [12, 18], 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.

In this paper we are going to propose a different approach to the problem. Our approach uses the framework of the Generalized Cascade (GC) model [3] to define the Inverse Infection Problem, the task of estimating the edge infection probabilities in diffusion or infection models. The problem can be reduced to an optimization task and a Particle Swarm based meta-heuristic is proposed to solve it. The Inverse Infection Problem and its solution is different from the previously mentioned approaches because it does not require information on the individual steps of the infection process. Instead it builds on the probabilities of infection of vertices both before and after the process itself.

In order to reduce the complexity of the task and allow the algorithm to handle large networks an additional simplification will be proposed. If additional information is available on both the vertices and edges of the graph in the form of attributes, attribute functions can be used to calculate the edge infection probabilities from the attributes themselves. The functions have unknown coefficients, and the task of the optimization is the estimation of these coefficients. It is also worthwhile to independently estimate the infection probability of each edge, although this is only possible in small graphs. This way we can shed more light on the behavior of the problem and we can explore the boundaries of the optimization algorithm. The usefulness of the Inverse Infection Problem has been proven in the recent publication of the authors in [1].

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. The paper is constructed as follows. In the next section we will give a short introduction into infection mechanisms, define the GC model, the Inverse Infection Problem and its two variants: direct and coefficient estimation. In Sect. 3 we will describe several ways to accurately estimate the infection probabilities, including gradient-based methods and Particle Swarm Optimization. We will

evaluate the performance of the optimization method on artificially generated infection scenarios in Sect. 4. The stability and the accuracy of the optimization will be examined, we will review the available infection heuristics and we will also identify the minimum number of patterns required to accurately estimate the infection probabilities. A preliminary version of this paper appeared in the proceedings of the Federated Conference on Computer Science and Information Systems [4].

2 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, [10] 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.

2.1 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 [11], 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.

2.2 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, 5] 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 Sect. 4, 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 [5]. 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 [7, 8], 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

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

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 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].

2.3 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 $e \in E(G)$.*

If we are looking for each edge weight independently, $|E(G)|$ severely limits the applicability of the method. It is possible to optimize for several thousand different values, but a graph with this amount of edges only fits into the small category. Even so, these small examples might provide us with important information about the problem and its resolvability.

To make the problem tractable for large real-life networks we are going to suggest simplification of the problem. In many real-life applications the probability of infection between vertices can be considered as a combination of some properties of the edges, vertices and the graph itself. We are going to take advantage of this fact to make the problem solvable on large graphs 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]

[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.

If there is only one attribute, this function can be expressed as $f(a_1(e))$, where f is the attribute function and $a_1(e)$ represents the attribute of edge e. We have used low-degree polynomials or simply a linear functions like $c_0 + c_1 a_1(e)$, where c_0 and c_1 are unknown coefficients. 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. If there are multiple attributes it is necessary to summarize the effect of them, and even in the case of a single one, normalization is required to get valid probability values between zero and one. Therefore it is necessary to use two functions, the attribute function is applied to the individual attributes on each edge, then the results of these functions are summarized and normalized: $w_e = g(f(a_1(e)), f(a_2(e)), \ldots, f(a_n(e)))$, where w_e is the edge weight of e, g is the summarizer-normalizer function, f denotes the attribute function and $a_i(e)$ represents the i-th attribute of edge e. The attribute and the normalizer function is the same for all edges, this way we only have to estimate the *coefficients* of these functions, and since the number of attributes and coefficients is limited, the problem becomes tractable.

According to the above we are going to define two variants of the problem.

1. The *direct estimation* of edge weights.
2. The attribute function *coefficient estimation*.

We are going to evaluate the properties of both variants in the results section.

3 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 choosing some initially random edge weights or coefficients and taking the a priori distribution we can compute an estimation of the a posteriori distribution. 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 by repeatedly adjusting the edge weights or coefficients and computing a new estimation of the a posteriori distribution. This is a typical task for global optimization, i.e. finding the minimum of an unknown multidimensional surface, where the points of this surface can be accurately estimated.

3.1 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 and the attribute formulation discussed above. While this early approach provided some results [5], it was clear that the search method was unable to solve but the most trivial tasks. 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.

Perhaps the most obvious problem is the dimensionality. Even if we consider coefficient estimation, the number of attributes and thus the number of parameters to estimate is too large for grid search to handle.

Another 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 third 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 Fig. 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 problem, and it also scales exponentially with the number of coefficients. The simple 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

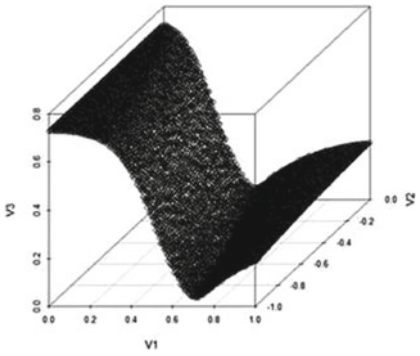


Fig. 1 The error surface of an IIP with one edge attribute and $f_1(a_1) = c_0 * a_1 + c_1$ as the attribute function. Root mean squared error was used for the evaluation

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

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

3.2 Particle-Swarm Optimization

In order to handle the above mentioned problems, we have decided to implement the particle swarm optimization algorithm of Kennedy [14]. This is an iterative method based on the interaction of multiple agents or "particles". Each agent corresponds to a different edge weight or coefficient configuration, representing a coordinate in the parameter space of the problem.

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 to it. 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 [15], and found, that it performs well in finding configurations close to the global minimum.

Algorithm 1 Particle Swarm Optimization

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

We have used the Fully Informed Particle Swarm published in [15] with a von Neumann neighborhood.[4] The position and the velocity of the agents are updated according to the following equations:

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

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

where $\mathbf{x}_i$ and $\mathbf{v}_i$ denotes the coordinate and velocity of particle i, $U(min, max)$ is a uniform random number generator, $\mathbf{b}_i$ 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 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 Eqs. 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.

4 Evaluation on Artificial Infection Scenarios

In this section we will test the performance of the Inverse Infection Problem and its estimation on artificially created infection scenarios. These scenarios take place on graphs with sizes from $|V(G)| = 1000$ to 100000. We will give a detailed discussion of the scenarios in Sect. 4.1. In most subsequent sections of the evaluation we will make a distinction between the direct and coefficient estimations and present result for both of them For the latter approach, a description of the used attributes and attribute functions will be given in Sect. 4.1.1.

We will begin the evaluation of the Inverse Infection Problem and its estimation method by testing the stability and running time of the optimization algorithm. We can show, that the latter can be divided into two parts: the evaluation of the error surface and the speed of the optimization mechanics. Corresponding to this we will start with the review of the complexities of the used infection heuristics and only then we will test the stability of the optimization. The most natural was to do this is by counting the average and maximum number of iterations the method takes before it finishes. In attribute estimation the number of values to estimate in our examples is at

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

most 21, while in direct estimation there can be thousands of values. Understandably the focus is on the latter.

The next point of interest is the effect of the infection heuristic on the precision of the estimation. Our findings correspond with those in [2], we will elaborate on this in Sect. 4.3. The results are mostly the same for each estimation type.

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. We will show the behavior of our method when the number of learning patterns is minimal. 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. We will discuss this in Sect. 4.4.

4.1 Test Dataset

As a basis of our analysis we have used graphs generated with the forest fire method of Leskovec et al. [19]. We have 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 the next subsection. 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 Complete Simulation with sample size $k = 10000$, because this method gives the best approximation of the original IC model.

4.1.1 Attribute Functions

We have seen, that in coefficient estimation, 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

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

where $\mathbf{e}$ 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 [5]. 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: $c_1 a_1(e) + c_2 a_2(\text{e})$, two coefficients in total.
- Weighted sum of four attributes: $\sum_i c_i a_i(e), i = 1, 2, 3, 4$, four coefficients in total.
- Weighted sum of six attributes: $\sum_i c_i a_i(e), i = 1, \ldots, 6$, six coefficients in total.
- Weighted sum of eight attributes: $\sum_i c_i a_i(e), i = 1, \ldots, 8$, eight coefficients in total.
- Weighted sum of ten attributes: $\sum_i c_i a_i(e), i = 1, \ldots, 10$, ten coefficients in total.
- Sum of quadratic polynomials with eight attributes $c_1 + \sum_i (c_{2i} a_i(e)^2 + c_{2i+1} a_i(e))$, $i = 1, \ldots, 8$, 17 coefficients in total.
- Sum of quadratic polynomials with ten attributes $c_1 + \sum_i (c_{2i} a_i(e)^2 + c_{2i+1} a_i(e))$, $i = 1, \ldots, 10$, 21 coefficients in total.

4.2 Computational Time

The time complexity of the inverse infection estimation 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 in Sect. 3.2, after initializing the neighborhood and the starting coordinates. Since the number of agents is relatively small, this part of the algorithm is very fast, and most of the running time is spent with the evaluation of error surface.

In each iteration every agent evaluates the error function. This evaluation is the computation of the GC model using the coordinates[5] of the given agent. The time complexity of this step heavily depends on the used heuristic. Altogether, we can

[5] Again, these are the edge weights or the coefficients of the attribute function.

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 different factors makes sense because of another reason: the individual runs of the infection heuristics may be run on multiple threads simultaneously, significantly improving the speed of the method. In the subsequent subsections we will discuss all of these starting with the evaluation of the error surface—the speed of the infection heuristics, the stability of the optimization algorithm and finally we will give a short example on the total running time of our method on an average personal computer.

4.2.1 Speed of the Heuristics

Previously, in Sect. 2.2, we have given short descriptions of some heuristics of the GC model published in [2]. We will give a short reminder on their running times on the networks discussed in the previous section. The mentioned heuristics are:

- 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 DeGroot [9] based simplification of the infection process.

We are going to use two different datasets to evaluate the running time. 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.

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 the heuristic to scale upwards and handle real-life datasets and networks with possibly millions of nodes and edges (Figs. 2 and 3).

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

[7] Note the sample sizes.

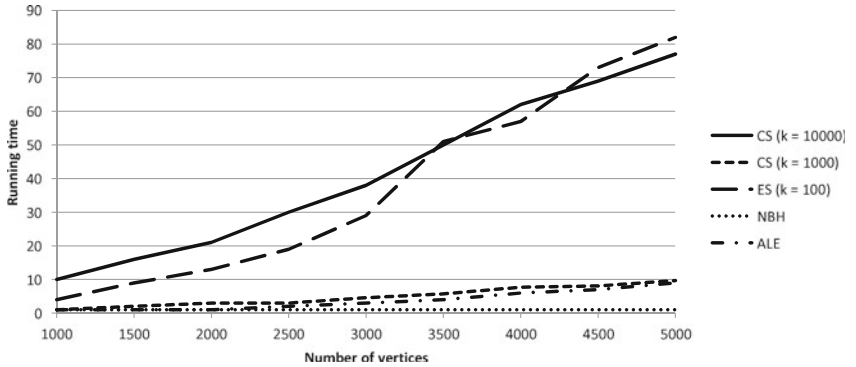


Fig. 2 The running time of the infection heuristics with different network sizes measured in seconds. Figure used with the permission of the authors [2]

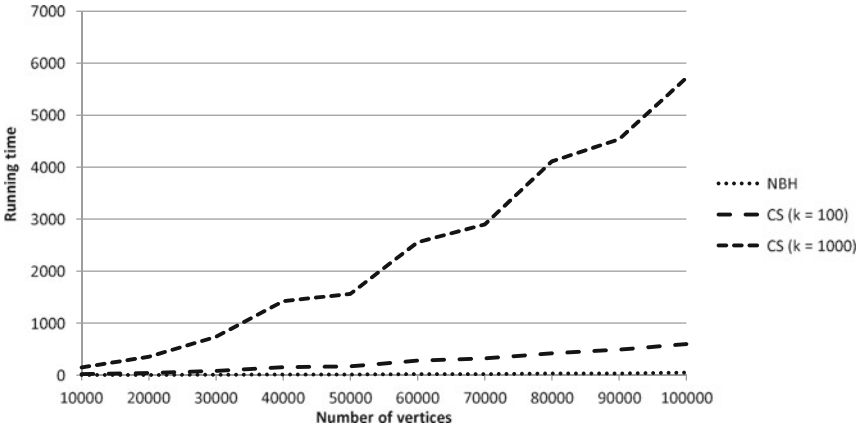


Fig. 3 The running time of the infection heuristics on large networks measured in seconds

4.2.2 Stability of the Optimization

Let us take a look at the stability of the optimization. We have to make a distinction between direct and coefficient estimations, because both the number of agents required to handle the task and the expected number of iterations are different.

On Fig. 4 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 Fig. 4 were computed with 9 agents. We have tried this problem with 16 agents as well and got similar results.

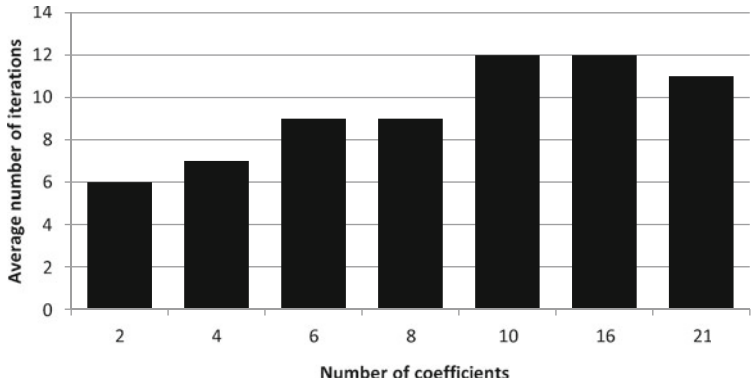


Fig. 4 The average number of iterations with different attribute functions

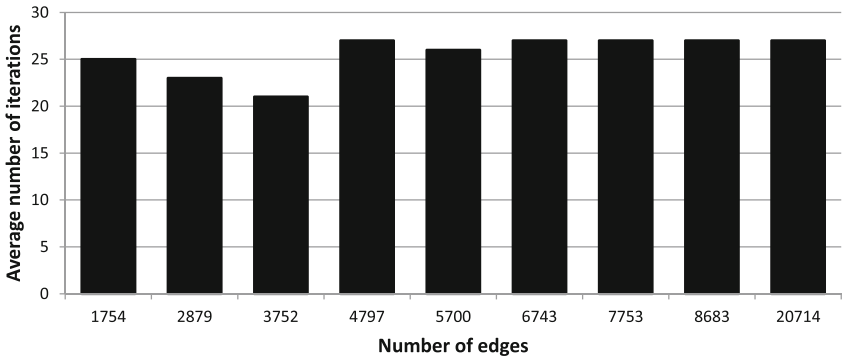


Fig. 5 The average number of iterations with direct estimation

If we compare the different infection heuristics, they perform similarly, with the non-Monte Carlo methods finishing slightly sooner, usually by 4–5 iterations.

A more interesting question is the stability of direct optimization, where dimensionality of the problem is in the thousands. On Fig. 5 the magnitude of the optimization task is represented as the number of edges $|E(G)|$ which is of course is equal to the dimensionality of the problem. The values correspond to networks with sizes of $|V(G)| = 1000, \ldots, 5000$. We can see that the average number of iterations is surprisingly low considering the task, it settles around 27, but the process rarely takes more than 50 iterations to conclude. The number of agents was 100. We have experimented with smaller and greater values, but they had little effect on the stability and the precision of the optimization.

Unlike before, there is a difference between the used heuristics. With this many agents, Edge Simulation is too slow to produce results even with $k = 100$. Complete Simulation performs well both in speed and accuracy, but with the other Monte Carlo method, the sample size should be significantly decreased. The Neighborhood Bound heuristic is very fast, but it fails to produce acceptable accuracy, as we will see in Sect. 4.3.

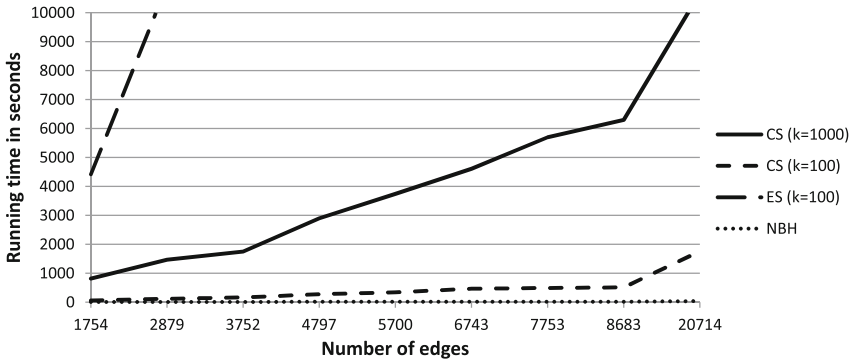


Fig. 6 Total running time of direct estimation

We can conclude, that the Particle-Swarm Optimization method described in this section is able to give an estimation of the Inverse Infection Problem with satisfying results. The algorithm is very stable, and even in the worst case of direct estimation, it finishes within 50 iterations.

4.2.3 Total Running Time

Finally let us take a look at the total running time of the algorithm. As we have mentioned before, the total running time is the function of the infection heuristic, the number of agents and the number of iterations and computations can be simplified by evaluating the points of the error surface in a parallel fashion. Therefore Fig. 6 is simply an example meant to illustrate that our method is able to produce results on an average personal computer. We can see the running times of the more demanding direct estimation with a hundred agents, and eight threads. We can confirm our previous observations: Edge Simulation is too slow to be of any use, and even though NBH is extremely fast we will see, that its accuracy is not acceptable. This leaves us with Complete Simulation.

4.3 Accuracy of the Inverse Infection Estimation

We are going to continue our discussion with the precision of the inverse infection estimation itself: we are going to test both direct and coefficient estimations on the artificial infection scenarios of Sect. 4.1. For our computations we will use the infection heuristics described in the previous section with the same settings. As we have mentioned before, we will use CS with sample size $k = 10000$ to create an a posteriori distribution as a reference set. Then we will use each heuristic with the optimization method to estimate the edge infection probabilities.

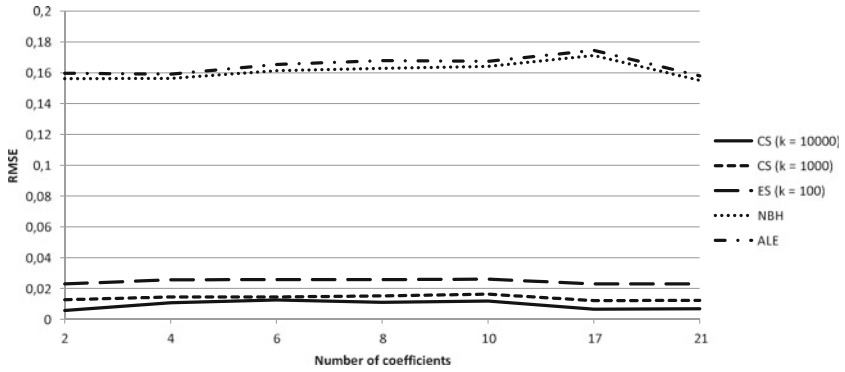


Fig. 7 The RMSE of coefficient estimation on a small network with $n = 1000$

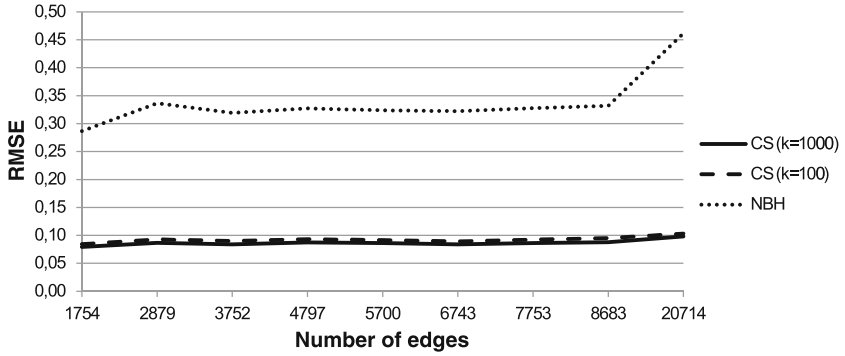


Fig. 8 The RMSE of direct estimation on networks with $|V(G)| = 1000, \ldots, 5000$

We can see the difference between the estimated and reference a posteriori infection values for coefficient estimation on Fig. 7. We have used a small network with $|V(G)| = 1000$. As we can see in this example, the Monte Carlo based simulations (CS and ES) are able to estimate the reference distribution well, with the measured error between 0.01 and 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. Finally, if we compare the results computed with different attribute functions, we can see that they have minimal effect on the accuracy of the methods. Yet this is only coefficient estimation so this is not surprising.

We can see the precision of direct estimation on Fig. 8. Again, the dimensionality of the task is represented as the size of $|E(G)|$ for graphs with $|V(G)| = 1000, \ldots, 5000$. Edge Simulation was left out because of the computational reasons discussed in the previous section, and ALE failed to produce acceptable output. We can see, that even though the Neighborhood Bound Heuristic is very fast, in direct

estimation it is unable to solve the optimization task. The only working heuristic is Complete Simulation, and even though it produces less accurate output than with coefficient estimation its precision is acceptable, with the added property of the accuracy remaining mostly the same as the dimensionality increases.

Considering both problem variants, we can conclude, that the use of Complete Simulation is recommended. Both its precision and accuracy are good for both direct and coefficient estimations. If the infection probabilities are lower than our current dataset, the use of Edge Simulation is also advisable for coefficient estimation. The Neighborhood Bound Heuristic is able to handle even larger networks, yet this comes at the cost of a significantly lower precision which only makes it applicable for coefficient estimation.

4.4 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.

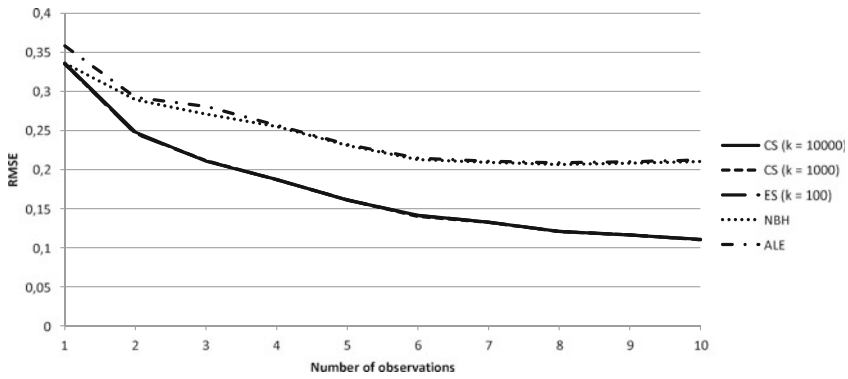


Fig. 9 The precision of the infection heuristics with a limited number of observations of the reference distribution

Different infection heuristics are shown on Fig. 9, one can see, that the simulations have identical performance regardless of the sample size. As before, the non-Monte Carlo based methods perform poorly, their use is not recommended with low-quality inputs.

5 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 and the Inverse Infection Problem. We have given two variants of the latter. The first one is the direct estimation of the edge weights of the given graph. We have also proposed coefficient estimation, where the edge weights are considered to be a function of known attributes on the edges, thus reducing the dimensionality of the problem.

We have formulated the Inverse Infection Problem as an optimization task and given a Particle-Swarm Optimization algorithm capable of giving a good estimation. Several aspects of the method were investigated: We have tested the stability and accuracy of the optimization method, 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 results were given for both direct and coefficient estimations. For both variants, the optimization method is able to 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. There are differences between the two variants however. It is only possible to use direct estimation if the graph is small enough, not just because the dimensionality of the problem, but because the number of agents required. This means the slower heuristics like Edge Simulation should not be used. The precision of the heuristics should be high, leaving NBH and ALE out of the picture, with Complete Simulation remaining to be the only recommended heuristic. Even with Complete Simulation, the accuracy of the method is less than with coefficient estimation. Still, if there are no attributes available and the graph is small enough, direct estimation can be used.

Coefficient estimation significantly reduces the dimensionality of the problem. It is faster, since less agents are required, and is less dependent on the used heuristic. Precision is the highest with the Monte Carlo heuristics, and even the Neighborhood Bound heuristic is able to produce acceptable results. The latter heuristic is extremely fast, with it is possible to handle large graphs with millions of nodes and edges. Coefficient estimation also handles low-quality inputs well.

We have shown the usefulness of our method in a recent paper, where we have given an application of this method in the prediction of credit default on bank transaction networks [1]. We were able to predict the bankruptcies of the riskiest 5 % clients 10 % more accurately than the currently used method. Our model was implemented in August 2013 into the OTP Bank of Hungary's credit monitoring process.

Acknowledgments 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, A. Csernenszky, L. Győrffy, G. Kovács, M. Krész, A. Pluhár, Applications of the inverse infection problem on bank transaction networks. Cent. Eur. J. Oper. Res. 1–12 (2014)
2. A. Bóta, M. Krész, A. Pluhár, Approximations of the generalized cascade model. Acta Cybern. **21**, 37–51 (2013)
3. A. Bóta, M. Krész, A. Pluhár, Systematic learning of edge probabilities in the Domingos-Richardson model. Int. J. Complex Syst. Sci. **1**(2), 115–118 (2011)
4. A. Bóta, M. Krész, A. Pluhár, The inverse infection problem, in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems* (IEEE, 2014), pp. 75–83. http://dx.doi.org/10.15439/978-83-60810-58-3
5. 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
6. T. Cao, X. Wu, T.X. Hu, S. Wang, Active learning of model parameters for influence maximization, in *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, ed. by D. Gunopulos, et al. (Springer, Berlin, 2011), pp. 280–295. doi:10.1007/978-3-642-23780-5_28
7. W. Chen, Y. Yuan, L. Zhang, Scalable influence maximization in social networks under the linear threshold model, in *Proceeding ICDM'10 Proceedings of the 2010 IEEE International Conference on Data Mining* (IEEE Computer Society, 2010), pp. 88–97. doi:10.1109/ICDM.2010.118
8. 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* (ACM, 2010), pp. 1029–1038. http://doi.acm.org/10.1145/1835804.1835934
9. M.H. DeGroot, Reaching a consensus. J. Am. Stat. Assoc. **69**(345), 118–21. http://www.tandfonline.com/doi/pdf/10.1080/01621459.1974.10480137
10. O. Diekmann, J.A.P. Heesterbeek, Mathematical epidemiology of infectious diseases, *Model Building, Analysis and Interpretation* (Wiley, Chichester, 2000)
11. P. Domingos, M. Richardson, Mining the network value of costumers, in *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining* (ACM, 2001), pp. 57–66. http://doi.acm.org/10.1145/502512.502525
12. A. Goyal, F. Bonchi, L.V.S. Lakshmanan, Learning influence probabilities in social networks, in *Proceedings of the Third ACM International Conference on Web Search and Data Mining* (ACM, 2010), pp. 241–250. http://doi.acm.org/10.1145/1718487.1718518
13. M. Granovetter, Threshold models of collective behavior. Am. J. Sociol. **83**(6), 1420–1443 (1978). http://psycnet.apa.org/doi/10.1086/226707
14. J. Kennedy, Particle swarm optimization. Encyclopedia of Machine Learning (Springer, US, 2010), pp. 760–766. doi:10.1007/978-0-387-30164-8_630
15. J. Kennedy, R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **36**(4), 515–519 (2006). doi:10.1109/TSMCC.2006.875410

16. D. Kempe, J. Kleinberg, E. Tardos, Maximizing the spread of influence though a social network, in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2003), pp. 137–146. http://doi.acm.org/10.1145/956750.956769
17. D. Kempe, J. Kleinberg, E. Tardos, Influential nodes in a diffusion model for social networks, in *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)* (Springer, 2005), pp. 1127–1138. doi:10.1007/11523468_91
18. M. Kimura, K. Saito, Tractable models for information diffusion in social networks, *Knowledge Discovery in Databases*, Lecture Notes in Computer Science (Springer, Berlin, 2006), pp. 259–271. doi:10.1007/11871637_27
19. J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations, in *Proceedings of the 1st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2005), pp. 177–187. http://doi.acm.org/10.1145/1081870.1081893



http://www.springer.com/978-3-319-21132-9

Recent Advances in Computational Optimization
Results of the Workshop on Computational Optimization
WCO 2014
Fidanova, S. (Ed.)
2016, X, 244 p. 90 illus., 15 illus. in color., Hardcover
ISBN: 978-3-319-21132-9