Acta Cybernetica **00** (0000) 1–15.

Approximations of the Generalized Cascade Model

András Bóta*, Miklós Krész† and András Pluhár‡

Abstract

The study of infection processes is an important field of science both from the theoretical and the practical point of view, and has many applications. In this paper we focus on the popular Independent Cascade model and its generalization. Unfortunately the exact computation of infection probabilities is a #P-complete problem [8], so one cannot expect fast exact algorithms. We propose several methods to efficiently compute infection patterns with acceptable accuracy. We will also examine the possibility of substituting the Independent Cascade model with a computationally more tractable model.

Keywords: computer science, infection process, heuristics

1 Introduction

The study of infection processes has many roots in various fields of research. The idea comes from the medical science for the purpose of modeling the spread of epidemics [13]. Boguña et al. [2] combined the infection processes with the emerging theory of Small World graphs, and gained new insights considering the survival of virulent diseases. Similar infection models can describe the spread of a behavior in social networks. One of the earliest models in sociometry, Granovetter's Linear Threshold model has proven to be an accurate description of information diffusion, see [15]. An important model in economics was developed by Domingos and Richardson [12] for the purpose of viral marketing. A form of the Independent Cascade model was later turned out to be an equivalent form of Granovetter's Linear Threshold model, see Kempe et al. [16, 17]. The exact computation of the vertex infection probabilities is a #P-complete problem [8], so applications are mainly using heuristics. Nevertheless, the IC model was also adopted to many other applications including costumer churn and the spread of credit default, [11, 4].

In later applications generalized IC models were used, in which the vertices become infected according to a general *a priori* probability distribution, before infecting their neighbors. Moreover, in these models an "infection" does not necessarily mean infection in the original sense, e.g. a bankruptcy of a company may

*University of Szeged E-mail: bandras@inf.u-szeged.hu
†University of Szeged E-mail: kresz@jgypk.u-szeged.hu
‡University of Szeged E-mail: pluhar@inf.u-szeged.hu

be caused by the poor economic state of its partners, but one cannot pinpoint the most responsible infector [19].

The initial problem described by Domingos and Richardson was influence maximization, that is to find a specific set of k individuals for a given $k \in N$ yielding the largest expected infection. Kempe et al. [16] proved that the exact solution is an NP-hard problem, but a greedy algorithm results in a k element set with expected influence at least $(1 - 1/e)Opt_k$ where Opt_k is the maximum influence for all set of size k. Still, a considerable amount of effort was spent to improve both the quality of the solution and the speed of the heuristics, [7, 8].

The computation of the infection process requires an input graph with edge weights corresponding to infection probabilities. However, in most practical applications the edge weights are unknown a priori, thus they are randomly generated or guessed. A more systematic approach appeared in the paper by Bóta et al. [5], in which the authors have proposed the *inverse infection problem.* That is, given the set of initial infectors (or a priori infection probability distribution in the general case), the output of the infection model (a posteriori infection probability distribution) and a graph structure, can we compute the infection probabilities? A different approach for the Linear Threshold model was developed by Cao et al. [6].

Both the infection maximization and inverse infection algorithms are based on the repeated computation of the IC model. Thus, the fast computation of this model is a requirement for any algorithm that tries to solve the above mentioned problems.

There are several existing algorithms for this purpose, each focusing on a different aspect of the model. We propose new methods, motivated by our observations on real economical data [11, 4]. These networks function fundamentally differently than social or interaction networks. More precisely, the spreading of credit defaults is quite different compared to the behavior of influenza or information diffusion. In general, a single defaulted company has little effect on its neighbors, since most companies have a wide array of dependencies both on business partners, customers and subcontractors. However, if a large number of companies go default at the same time, the effect on other companies becomes gradually greater. According to this, on the examined networks the edge infection probabilities are low, typically below 0.2, or even smaller.

In this paper we are going to describe three methods exploiting the above aspect of the problem.

- The Edge Simulation method is a combination of both simulation and exact computations that decreases the standard deviation appearing in other simulations.
- If the infection probabilities are small, then the infections typically do not travel far from the source of infection. Neighborhood Bound Heuristics exploits this property.
- The Independent Cascade model itself can be substituted for a similar, but a computationally more tractable model.

The rest of this paper is organized as follows. First we define the original IC model and generalize it to suit complex models. We will also give a short description of the inverse infection problem. Then we describe the above mentioned methods in detail. Finally we compare the results with respect to the speed and accuracy of the computations.

2 Problem definition

In this section we will define the Independent Cascade model, the Generalized Cascade model and the inverse infection problem.

The pair $G = (V, E)$ is a directed graph, where V denotes the set of nodes and $E \subset V \times V$ denotes the set of directed links. If there is a $w_{u,v}$ defined for each edge (u, v), we have a weighted graph. When dealing with infection models, we restrict the values of the weights to be $w_{u,v} \in [0, 1]$ for each edge (u, v). We will also refer to the weights as infection probabilities.

The *Independent Cascade model* is an iterative method based on the process of active nodes infecting inactive ones. The process starts with an initially active set of nodes $A_0 \subset V(G)$. Let $A_i \subseteq V(G)$ be the set of nodes newly activated in iteration i. In iteration $i + 1$, every node $u \in A_i$ has one chance to activate each of its inactive out-neighbors $v \in V \setminus \cup_{0 \leq j \leq i} S_j$ according to $w_{u,v}$. If the attempt is successful, then v becomes active in iteration $i+1$. If more than one node is trying to activate v in the same iteration, the attempts are made in an arbitrary order and independently of each other still in iteration $i + 1$. The process terminates at step t if $A_t = \emptyset$.

The *Generalized Cascade model* extends this process in the following way. Each $v \in V(G)$ has an *a priori infection probability* w_v, and the vertices become active independently of each other with their assigned probabilities at the beginning. After this, the infection process of the IC model applies with a randomly selected active set. In this process, each vertex gets an *a posteriori infection probability* w'_v corresponding to the probability of infection at the end. In other words, the process can be interpreted as a transformation of probability distributions over the graph G.

The Generalized Cascade model can be summarized as follows: Given a graph G, the edge infection probabilities $w_{u,v}$ for all $(u, v) \in E(G)$, and the a priori probabilities w_v for all $v \in V(G)$, we are looking for the a posteriori infection probabilities w'_v for all vertices $v \in V(G)$.

All these applications of infection models require the edge probabilities to be given beforehand. However, in practice these values are unknown. Therefore it is necessary to develop an algorithm that is capable of *learning* these weights. That is, if we have the a priori and a posteriori distributions of the GC model as the input of an algorithm, it should assign edge weights $w_{u,v}$ for all edges (u, v), such that the a priori distribution is transformed into a good approximation of the a posteriori distribution. Some algorithms for the above problem were investigated in [5]. Note that all of these methods require the repeated computation of the

Generalized Cascade model.

The exact computation of the vertex infection probabilities is difficult [8]. Some authors have tried to circumvent this by developing algorithms for special classes of graphs, like trees or DAG-s [7]. In this paper, we will describe four methods for approximating the a posteriori infection probabilities in the GC Model.

3 Methods

The methods presented below fall into three categories. *Simulations* are Monte Carlo generators, i.e. they compute multiple realizations of the probabilistic infection process, and count the relative frequency of vertex infections.

In contrast to this, *heuristics* use an approximation of the GC model, circumventing the #P-completeness, but still dealing with both the a priori and the edge infection probabilities. Finally, *hybrid methods* use some combination of both approaches.

3.1 Frequency based simulations

The notion of using Monte Carlo based simulations to compute the a posteriori infection distribution comes from Kempe et al [17]. That is, generating random edges and vertex infections, according to the edge weights and the a priori vertex infections, and then approximate the infection probability of vertices by the relative frequency of vertex infections.

All methods described in this section use the above approach. The original method of Kempe et al. can be adapted to the requirements of the Generalized Cascade model. We will refer to this as **Complete Simulation**. In contrast to this, **Edge Simulation** is a hybrid design using heuristics to improve the performance of the simulation.

3.1.1 Previous works

Kempe, Kleinberg and Tardos developed a method based on reachability [17] to compute the Independent Cascade model. They construct an unweighted directed graph G' on the same vertex set as G by drawing the edge (u, v) independently of the other edges with probability $w_{u,v}$. The resulting graph G' can be interpreted as a realization of the possible routes of infection. Those vertices that can be reached from an initially infected vertex also become infected. This way the computation of the iterations one by one can be avoided, and the problem of a single simulation step is reduced to a simple (path) searching problem. Note that this formulation helped to show that $f(A)$, the expected infection of a set A is a *submodular function*, that is $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$ for all $v \in V(G)$ and $S \subset T \subset V(G)$.

The process can be applied to G multiple times, resulting in a series of realizations. The desired distribution can be approximated simply by counting the relative frequency of vertex infections.

3.1.2 Complete Simulation

It is easy to adapt the method of Kempe et al. to the needs of the Generalized Cascade model. In addition to the construction of G', a set $A_0 \subseteq V(G)$ is created, according to the following rule: for all $v \in V(G)$, $v \in A_0$ according to the probability w_v. G' and A_0 is generated multiple times. We will refer to this value as the sample size k. Finally let f_v denote the relative frequency of infection for vertex v.

Algorithm 1 Complete Simulation

Input: Graph G, sample size k
Output: Relative frequency of infection f_v for all $v \in V(G)$

```
1:  j ← 0
2:  for all v ∈ V: f_v ← 0
3:  while j < k do
4:     Generate G'
5:     Generate A_0
6:     S ← ∅
7:     Q ← V(G')
8:     for all u ∈ A_0 ∩ Q do
9:        for all v ∈ Q do
10:          if there is a path u, ..., v in graph G' then
11:             S ← S ∪ {v}
12:          end if
13:       end for
14:       Q ← Q \ S
15:    end for
16:    for all v ∈ S: f_v ← f_v + 1
17:    j ← j + 1
18: end while
19: for all v ∈ V: f_v ← f_v / k
```

At line 10, we allow $u = v$. In the main loop, it is indifferent which node in A_0 is the source of the infection. Since a node can be infected only once, any node reachable from a node from A_0 can be left out of further computations. Because of this, the for loops in the algorithm can be interpreted as a single search on G' starting from the nodes in A_0.

Before the main loop, f_v is initialized for all vertices. It is easy to see, that the algorithm computes the relative frequencies for all vertices correctly. Since this method is based on frequency counting, its precision and time complexity depends greatly on the sample size k. We will elaborate on this in Section 4.

3.1.3 Edge Simulation

There is a different way to simulate the generalized model: instead of computing A_0 we can deal directly with the a priori vertex infection probabilities. The notation

is the same as before, w_v denotes the a priori infection probability of vertex v. At line 8, we allow $u = v$.

Algorithm 2 Edge Simulation

Input: Graph G, sample size k
Output: Relative frequency of infection f_v for all $v \in V(G)$

```
1:  j ← 0
2:  for all v ∈ V: f_v ← 0
3:  while j < k do
4:    Generate G'
5:    for all v ∈ V(G) do
6:      s ← 1
7:      for all u ∈ V(G) do
8:        if there is a path u, ..., v in graph G' then
9:          s ← s(1 − w_u)
10:       end if
11:     end for
12:     f_v ← f_v + 1 − s
13:   end for
14:   j ← j + 1
15: end while
16: for all v ∈ V: f_v ← f_v / k
```

In contrast to the previous method, the source of the infection does matter, since the value w_v is not the same for different vertices. We also do not have any restrictions on the structure of G other than being simple, so any vertex can be a part of a loop. This means, that in order to avoid counting a single w_v multiple times, we have to do a search for each node independently. Obviously the above approach increases the time complexity in general, but if the edge probabilities are low enough in G, then G' has many small components, reducing the running time for each search significantly.

It is clear, that the computational time is greater for ES compared to CS. In exchange, we can expect, that the precision of the approximation is less dependent on the sample size. We will elaborate on this in Section 4.

3.2 Neighborhood Bound Heuristic

There are several existing heuristics for the IC model [7, 8, 10, 9] covering a large area of performance requirements. These methods usually exploit one or more properties of the infection process. Chen's DAG and LDAG algorithm for example focuses on the concept, that the edges with high infection probabilities carry the bulk of the infection process. They propose to construct a local directed acyclic graph containing the relevant edges for each node, and then compute the infection process using the DAG.

The main idea of our method is similar: the construction of a small graph containing all of the possible paths of infection inside a given neighborhood for a given vertex. This graph is created in such a way, that the approximation of the a posteriori infection probability of the corresponding vertex can be easily computed. It is important to emphasize, that the goal of this method is the approximation of the GC model as fast as possible, disregarding requirements for precision.

We are also going to rely on the findings in [11, 4]. Based on the examination of economic networks, the authors have found, that the edge infection probabilities are small, typically below 0.2. The above observation greatly limits "the travel distance" of an infection event, since even if we consider a path of length two from the source of the infection, the probability of infection is reduced to 0.04 or below.

Based on the remarks above, we propose the Neighborhood Bound Heuristic (NBH). We will denote the set containing the in-neighbors of vertex v as $N^-(v)$. For all $v \in V(G)$ we are going to construct a weighted, rooted tree T_v with v as the root, and edges pointing towards v. In the first step all vertices $u \in N^-(v)$ are added to T_v, as well as the edges $(u, v) \in E(G)$ for all $u \in N^-(v)$. In the second step we are going to deal with the second neighborhood of v. For all $u \in N^-(v)$ we are going to add the nodes $z \in N^-(u) \setminus \{v\}$ and all of the edges $(z, u) \in E(G)$ to T_v. The subtractions of v is necessary to avoid loops of length two. For all edges in T_v we keep the edge weights from G. Take note, that nothing prohibits the nodes of G (with the exception of v) from appearing multiple times in T_v, this corresponds with our idea of representing all possible infection paths in the second neighborhood of v.

The computation of the a posteriori infection probability of v in T_v is easy. T_v has three levels: the leaves, $N^-(v)$ and the root v. The a posteriori infection probabilities of the leaves are the same as their a priori ones, since they do not have in-neighbors. A node $u \in N^-(v)$ gets infected if one of the leaves connected to it is infected, or becomes infected by itself, meaning $w'_u = 1-(1-w_u)\prod_{z \in N^-(u)}(1-w'_z)$. The computation is executed in the same fashion for v.

The above method is extremely fast, since w'_v can be computed in a single run on the edges of T_v, and if G is sparse, $|E(T_v)|$ is small. The construction of T_v is simple, if we limit the process to the second neighborhood of v in G. If we consider larger neighborhoods, nothing prohibits $|V(T_v)|$ from growing exponentially, and it becomes increasingly difficult to avoid loops. Finally, if we consider the fact, that the edge weights are small, then the loss of precision is still within acceptable bounds.

3.3 Aggregated Linear Effect Model

Our goal in this section is to build up a model that more or less approximates the mechanism of the Generalized Cascade model. We begin with some motivations and define the *Aggregated Linear Effect Model*, shortly ALE model afterward. For a weighted graph G, let the a priori infection of a vertex v be w_v. If one considers only one step of the linear effects at v, it can be defined as $\sum_{u:u \in N(v)} w_{u,v} + w_v$. This is nothing else, but $x + Bx$, where B is the transpose of A, the weighted

incidence matrix of G, and x is the vector of a priori infection probabilities.

In the 2nd, 3rd, ... ith steps we may aggregate the effects of the second, third etc. neighborhoods by adding the B^2x, B^3x, ..., B^ix correction terms to the approximation.

3.3.1 Definition of ALE model.

Let $B = A^T$, where A is the weighted incidence matrix of graph G, and x is the vector, such that $x_v = w_v$, then the a posteriori effect y is defined as

$$y := (I + B + B^2 + \dots)x,$$

where I is the identity matrix.

Note, that $\sum_{u:u\in N(v)} w_{u,v} + w_v$ is close to the probability of v getting infected independently in one step by itself or by a neighbor. In general, if the weights are small, the error is negligible. Assuming small weights, the infinite series also converges, and we have the more compact form of

$$y := (I + B + B^2 + \dots)x = (I - A)^{-1}x.$$

The values w_v and $w_{u,v}$ do not have to be probabilities any more, we might consider any appropriate scalar functions (perhaps after scaling in order to maintain convergence). Now the following questions arise:

How good is the ALE model? How to compute (and later on use) it efficiently?

There are two obvious ways to test the first question. One is to consider some real problems, make a network model out of those, and compute the optimal weights in an ALE model such that the model "prediction" are as close to the real values as possible. However, since the IC models already performed well in such case studies, we might consider here an easier way. We just take a weighted G with some a priori infection x, compute the a posteriori infection y by the associated IC model, and then try to find appropriate new weights such that for the a posteriori effect y' by the ALE model we get $\min ||y - y'||$ in a fixed norm.

Once we have the appropriate weights, that is the matrix B, we can compute y easily. Of course not by inverting $I - B$ but solving the equation $(I - B)y = x$ for y by Gaussian elimination.

4 Results

For the purposes of evaluation we have used sparse graphs generated with the forest fire model [18], with $|G(V)| = n = 1000, \dots, 40000$. This allows us to examine how much the computation time of a given algorithm scales with graph size. Since the performance of most methods described here depends on the size of the infection probabilities, we have used five arrangements of edge weights and a priori distributions.

Setup	A	B	C	D	E
ℓ_1	0.02	0.05	0.1	0.2	0.5
ℓ_2	0.1	0.1	0.2	0.2	0.5

Table 1: Experiment setups.

- Each edge weight is drawn independently from an uniform distribution between $(0, \ell_1)$.
- The expected size of the set of random infectors is $n * \ell_2$. For each vertex, there is an a priori infection probability drawn independently from an uniform distribution between $(0, 0.2)$.
- For all other vertices $w_v = 0$.

Using the above process, for each of the unweighted graphs, we have created five weighted ones with a priori probabilities, resulting in $9 \times 5 = 45$ different benchmark networks. The parameters ℓ_1 and ℓ_2 of each arrangement can be seen in table 1.

We have mentioned in the introduction, that our aim is the analysis of economic networks. We have also mentioned our findings, that infection probabilities in these networks are usually small. Based on this, we consider the probability arrangements A through D to be within our area of interest. Setup E is used to measure the behavior of described methods outside these conditions.

In this section, we are going to evaluate the running time and precision of our methods[1]. While the evaluation of computational time is rather straightforward, we have to make a distinction while measuring performance.

Simulations are Monte Carlo based estimations of the infection process. The goodness of an estimation is governed by the sample size k. From the Law of Large Numbers it follows that if k goes to infinity then the simulationed values converge to the infection probability. Depending on k, the output of the simulations differ from each other, characterized by their deviation. It is important to emphasize, that the deviation only depends on k. It does not depend on the graph size or the experiment setup.

On the other hand, heuristics apply a process similar to the infection model to approximate its output in reasonable time. The goodness of the approximation can be measured by comparing it with a simulation computed with a k large enough to minimize deviation. The difference between these can be described by an error function[2]. The error of these heuristics is highly dependent on the sizes of the edge weights as well as the size of the network.

Based on the above facts, we will evaluate the precision of the simulations and the heuristics somewhat differently.

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

[2] For this purpose we have used the root mean squared error function (RMSE).

4.1 Deviation of the simulations

The most important question of simulation is the relation between the sample size and the accuracy of the simulation. This can be measured as the standard deviation between the a posteriori infections [3]. Increasing k obviously worsens the running time of the simulation, so it is desirable to find a balance between accuracy and complexity.

Let us make some heuristics concerning the expected results of CS. The worst case for the variance is when the characteristic function X_v of the infection of a vertex v follows a Bernoulli distribution. Then the standard deviation of X_v is $\sigma_v = \sqrt{p_v(1-w_v)}$. The standard deviation of the k-element sample is

$$\frac{\sqrt{kw_v(1-w_v)}}{k} = O\left(\frac{1}{\sqrt{k}}\right),$$

that is to get one more correct digit in the outcome one needs one hundred times more iterations.

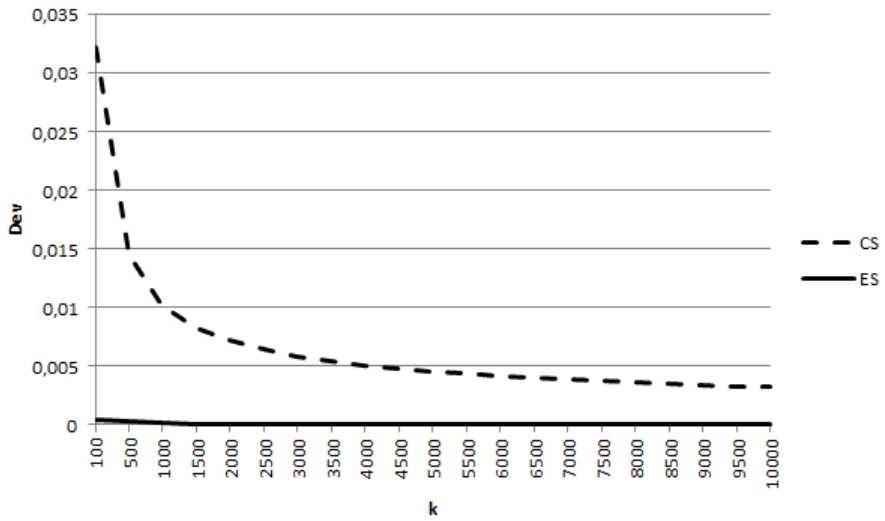

Figure 1: The absolute deviation compared to the sample size. In order to evaluate very large values of k, we have used a small benchmark network.

Of course, $X_v = Y_v + Z_v - (Y_v Z_v)$, where Y_v and Z_v are the characteristic functions of the a priori infection and the infection caused by the network, respectively. We might assume Y_v and Z_v independent of each other, and Y_v follows Bernoulli distribution. Now, if we simulate the edges and use the vertex a priori infection

[3]For testing purposes, we have used Zachary's karate club network [20] with edge weights between 0 and 0.5 and four initially infected nodes. The a priori infection probabilities of these nodes were drawn independently from an uniform distribution between 0 and 1.

probabilities directly, then the approximation of p_v is significantly improved. Indeed, an easy computations gives that

$$Var(X_v) = Var(Y_v) + Var(Z_v) \leq \mathbb{E}[Y_v](1 - \mathbb{E}[Y_v]) + Var(Z_v).$$

However, if we handle Y_v as a constant of value $\mathbb{E}[Y_v]$ then $Var(X_v)$ drops to $Var(Z_v)$. The infection coming for the network is usually much smaller than the a priori infection, and since it is the sum of almost independent variables, the variance of X_v must be even much smaller.

Figure 1 shows the standard deviation compared to the sample size k for both methods. The deviations are averaged for all nodes of the network. It can be seen, that the empirical results for CS roughly correspond to our heuristics. It can also be seen, that even for small values of k ES performs better by magnitudes than CS. In fact, the deviation goes below 10^{-5} for $k = 1500$, while using CS it goes below 10^{-4} only for $k = 100000$.

4.2 Precision of the heuristics

In order to measure the accuracy of NBH and ALE, we have used a very accurate Edge Simulation with $k = 5000$ as a benchmark. It can be seen above that the expected error of this simulation is less than 10^{-5}. Root mean squared error is used to measure the difference between the a posteriori distributions.

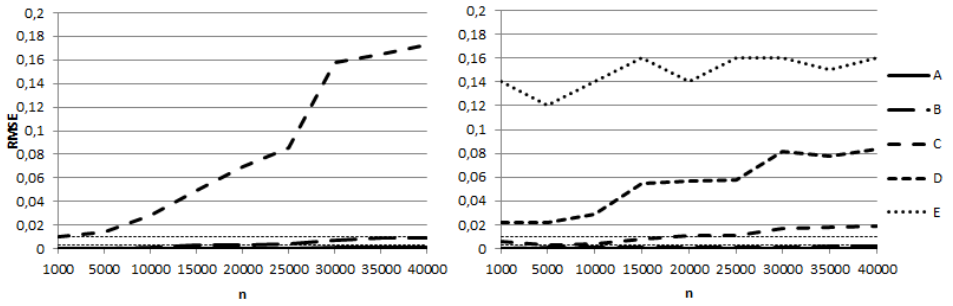

Figure 2: The RMSE of ALE (left) and NBH (right) for all experiment setups. The small dashed lines indicate the standard deviation of CS for $k = 10000$ and $k = 1000$.

On the left hand side of Figure 2, we can see the performance of ALE compared to the benchmark. If the edge weights are small enough (below 0.1), ALE is able to approximate the GC model with reasonable accuracy. For higher weights the performance of ALE gradually worsens to the point, that for the last two setups it is not able to produce output within acceptable bounds. The accuracy of the method also depends on the size on the graph, although for the first two setups, this is barely noticeable. The situation does not so grim if we consider our experiences

with the edge weights of economic networks, which typically fall into the first two category.

If we take a look at the performance of NBH on the right, we can see, that if the edge weights are below 0.2 (Setup D), the error remains below 0.1. It is easy to see, that lower edge infection probabilities produce more accurate approximations, which confirms our expectation, that if these probabilities are low enough, infections do not travel far. Like ALE, NBH is also slightly dependent on the size of the network. It is also clear, that NBH performs better than ALE in general, producing lower error levels for the same arrangements of infection probabilities.

We can compare these result to the deviation of the simulation based methods. The small dashed lines on both parts of Figure 2 indicate the standard deviation of CS with $k = 10000$ (lower) and $k = 1000$ (higher). Take note, that according to Figure 1, the deviation of ES is lower, even for $k = 100$. Based on the above fact, we can say, that the performance of both methods is comparable to the simulations only if the infection probabilities are small enough.

4.3 Computational time

On Figure 3 we can see the computational time of our methods on four probability arrangements. The results for setup A and B were almost the same, so we have left out the former one.

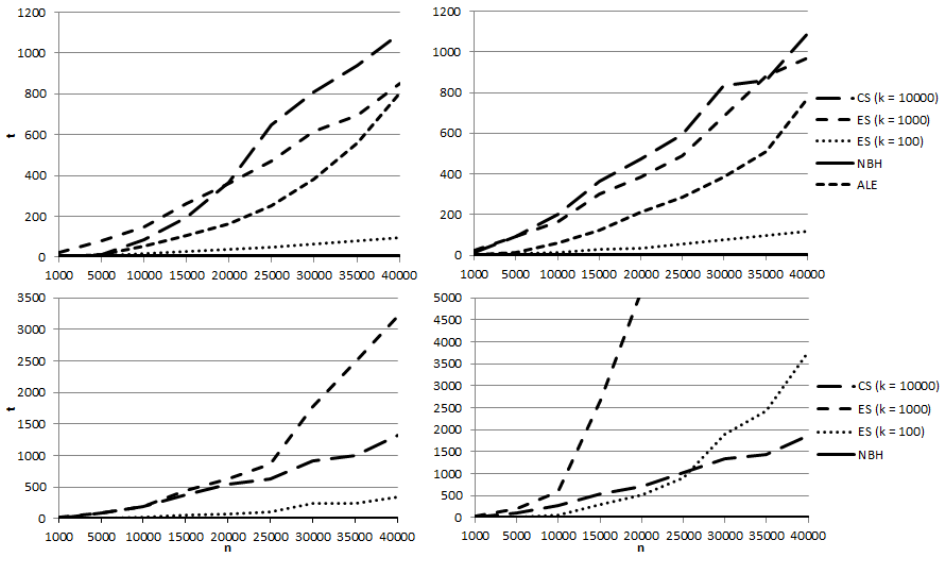

Figure 3: The running time of the algorithms measured in seconds, compared to the size of the graphs. Experiment setup B (upper left), setup C (upper right), setup D (lower left) and setup E (lower right).

The running time of CS scales more or less linearly with graph size, indepen-

dent of probability setups. However, the deviation of CS only decreases below an acceptable level if k is sufficiently high. This means, that even tough the computation of a single G' is fast, a large amount of realizations have to be generated in order to compete with other methods. As a consequence, CS is the slowest of the methods on the first three experiment setups. Its use is only recommended if the probabilities are too high for the other algorithms to tackle.

Unlike CS, the performance of ES is highly dependent on the size of infection probabilities. The computational time gradually worsens with the increase of edge weights, to the point where even $k = 100$ is infeasible. This corresponds with our remarks in Section 3.1.3. Take note though, that the deviation of ES is smaller than CS by magnitudes, meaning an ES with $k = 100$ outperforms a CS with $k = 10000$ in terms of precision. In our area of interest, ES seems to be a reasonable compromise between precision and computational time.

Due to its local nature, NBH is the fastest of the methods. Even on the largest graphs it completes the task in less than two seconds for all experiment setups. As we have seen in the previous section however, this comes at a cost of decreased precision, on all but the smallest probability arrangements. As a consequence, its use is only recommended if the network is large, and the infection probabilities are low.

ALE scales similarly compared to the simulations with a slight increase in performance for the first three probability arrangements, but it is unable to produce meaningful output on the last two setups. If we take the findings of the previous section into account, then we can conclude, that ALE should only be used if the infection probabilities are very low.

5 Conclusions

In this paper we have described four different methods able to compute the Generalized Cascade model. All of these methods are based on the observation, that in realistic networks, the edge infection probabilities are low.

The Edge Simulation is a Monte Carlo based method that greatly reduces the variance of the resulting a posteriori distribution. Tests indicate that the variance can be reduced by two magnitudes with the same sample size. Unfortunately the method requires more computations than other simulations, and its speed depends on the edge infection probabilities.

The Neighborhood Bound Heuristic limits the effect a node has on another one to a distance of two. Its accuracy is surprisingly good in graphs with small weights, but even in this case, care must be taken with its use.

The direct adaptation of the method developed by Kempe et al. can be an acceptable choice if the infection probabilities are high. The computational time of CS is independent of the former, and scales linearly with both the size of the network, and the number of samples.

The Aggregated Linear Effect is a model that tries to approximate the mechanism of the Generalized Cascade model, but is computationally more tractable

only if the edge weights are small enough. It is also important to note, that the inverse infection problem can be solved directly with the use of ALE, avoiding the additional cost of a learning algorithm.

Acknowledgments. The first and third authors were partially 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).

The second author was partially supported by the European Union and co-funded by the European Social Fund through project HPC (grant no.: TÁMOP-4.2.2.C-11/1/KONV-2012-0010).

The second author was also supported by the Gyula Juhász Faculty of Education, University of Szeged (project no. CS-004/2012).

References

[1] R. Albert, A. L. Barabási, Statistical mechanics of complex networks. *Reviews of Modern Physics*, **74** (2002) 47–97.

[2] M. Boguña, R. Pastor-Satorras, A. Vespignani, Absence of epidemic threshold in scale-free net-works with degree correlations. *Phys. Rev. Lett.* Vol. **90** No 2. (2003) 028701-1–4 .

[3] B. Bollobás, *Modern Graph Theory,* Springer, New York (1998).

[4] A. Bóta, L. Csizmadia and A. Pluhár, Community detection and its use in Real Graphs. *Proceedings of the 2010 Mini-Conference on Applied Theoretical Computer Science - MATCOS* **10** (2010) 95–99.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

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

[14] M.E.J. Newman, The structure and function of complex networks. *SIAM Review* **45**, (2003) 167-256.

[15] M. Granovetter, Threshold models of collective behavior. *American Journal of Sociology* **83**(6) (1978) 1420–1443.

[16] 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.

[17] 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.

[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.

[19] M. Krész and A. Pluhár, Prediction of Economic and Social Events by Infection Processes. To appear in *Encyclopedia of Social Network Analysis and Mining*, Springer (2012).

[20] W. W. Zachary, An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* **33** (1977), 452–473.

Received ...