Spreading Rumours without the Network

Paweł Brach[*]
University of Warsaw
pawel.brach@mimuw.edu.pl

Alessandro Epasto[†]
Sapienza University of Rome
epasto@di.uniroma1.it

Alessandro Panconesi[‡]
Sapienza University of Rome
ale@di.uniroma1.it

Piotr Sankowski[§]
University of Warsaw
sank@mimuw.edu.pl

ABSTRACT

In this paper we tackle the following question: is it possible to predict the characteristics of the evolution of an epidemic process in a social network on the basis of the degree distribution alone? We answer this question affirmatively for several diffusion processes– Push-Pull, Broadcast and SIR– by showing that it is possible to predict with good accuracy their average evolution. We do this by developing a space efficient predictor that makes it possible to handle very large networks with very limited computational resources. Our experiments show that the prediction is surprisingly good for many instances of real-world networks. The class of real-world networks for which this happens can be characterized in terms of their neighbourhood function, which turns out to be similar to that of random networks. Finally, we analyse real instances of rumour spreading in Twitter and observe that our model describes qualitatively well their evolution.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; G.3 [**Probability and Statistics**]: Stochastic processes

General Terms

Algorithms

[*]Partly supported by ERC StG project PAAl 259515, FET IP project MULTIPEX 317532 and NCN grant N N206 567940.

[†]Partially supported by a Google Europe PhD Fellowship in Algorithms and by the EU FET project MULTIPLEX 317532.

[‡]Partially supported by a Google Faculty Research Award, the CASPUR HPC Std Grant 2012 and by the EU FET project MULTIPLEX 317532.

[§]Partly supported by ERC StG project PAAl 259515, FET IP project MULTIPEX 317532 and NCN grant N N206 567940.

COSN'14, October 1–2, 2014, Dublin, Ireland.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3198-2/14/10 ...$15.00.
http://dx.doi.org/10.1145/2660460.2660472.

Keywords

rumour spreading; degree distribution; configuration model; social networks; push-pull; sir; neighbourhood function.

1. INTRODUCTION

In recent years, we have witnessed the emergence of many sophisticated web services that allow people to interact on an unprecedented scale. The wealth of data produced by these new ways of communication holds the promise to increase our understanding of human social behaviour, but a fundamental hurdle is posed by the sensitivity of these data. It could be available in principle, but its access must be severely constrained to protect the privacy of users and the confidentiality of data. In some contexts, e.g. sociological and epidemiological studies, the network topology for which the diffusion processes are to be evaluated might simply be unknown, or impossible to be determined with any accuracy. It is often possible, however, to gain relatively unconstrained access to some crude statistical information about the data. For instance, while the exact network topology of a social graph could be unavailable, some aggregate information such as its degree distribution can be disclosed and made public. A broad question arises naturally: can non-trivial conclusions about various social processes be inferred based only on such limited access to the data?

We tackle this question in the case of diffusion processes. We study some classical models of information diffusion (which we henceforth refer collectively as rumour spreading[1]) – gossiping in its PUSH and PULL variants, broadcast and SIR– and ask the following question: is it possible to predict the average evolution of the epidemic as it spreads in a given real-world network on the basis of its degree distribution alone? Furthermore, given the size of current data sets, we want to perform such a prediction in a space efficient manner, to make the prediction for very large graphs possible with limited computational resources. Note that on the basis of the degree information alone, we can only hope to predict how the process will evolve on average, for there can be graphs with the same degree distribution for which the epidemic will evolve at very different speeds.

Surprisingly, we show that our goal is achievable for a large class of real-world networks. This is the class of real networks whose neighbourhood function [6] closely resembles that of a random graph with the same degree distribution.

[1]We stress that, contrary to the common meaning of the word "rumour", in the context of the literature on information diffusion processes rumour simply means a generic information.

The above discussion gives the gist of our results. Before describing them in detail, let us review the diffusion models we will be considering.

Push, Pull and SIR.

Gossip (a.k.a. random-call model) is a fundamental communication primitive inspired by the dynamics of viral diffusion and word-of-mouth. It comes in three guises known as PUSH, PULL and PUSH-PULL. In each, the process starts at time $t = 0$ with one source node with a message. The process evolves in a sequence of discrete, synchronous rounds. In the PUSH variant, in round $t \geq 0$, every informed node selects a neighbour uniformly at random to which the message is forwarded. The PULL variant is symmetric, i.e., in round $t \geq 0$, every node that does not yet know the message selects a neighbour uniformly at random and asks for the information to be transferred, provided that the queried neighbour has it. The PUSH-PULL variant is a combination of both, in each round informed nodes execute a PUSH and uniformed ones a PULL.

From a computer science perspective these models provide simple-to-implement tools for the dissemination of information across a network [12] and for this reason they have been studied extensively in practice and theory. More generally, they provide a very simple model for the dynamics of viral diffusion that, in spite of their simplicity, can offer some insight into more realistic processes in sociology [26], economics [21], and epidemiology [20]. There is a rich theoretical literature the studies the speed of these protocols. To draw a few paradigmatic examples from a rich body of literature, Feige *et al.* [13] showed that, in any connected network, PUSH delivers the message to every node within $O(n \log n)$ many rounds, where n denotes the number of nodes. These results have been extended to PUSH-PULL over random graphs with power law degree distribution with exponent α in [14]. It has been shown that when $2 < \alpha < 3$ the rumour spreads to almost all nodes in $O(\log \log n)$ time, whereas when $\alpha > 3$ one needs $\Omega(\log n)$ rounds. In a series of relatively recent papers [11, 10, 15] a tight relationship was established between the speed of diffusion of PUSH-PULL and the conductance of the network, denoted as φ, culminating with the optimal $O(\log n/\varphi)$ bound [15]. Notice that all these results are asymptotic while our focus here is different. We want to predict the expected number of informed nodes at time t, on the basis of the degree distribution of the network.

In this paper we also evaluate a classical diffusion model inspired by mathematical epidemiology studies [20, 16], the SIR process. In this process, a node in the network can be either *susceptible*, *infected* or *removed*. At time $t = 0$ a single node is infected and all the others are susceptible. At time $t \geq 0$, infected nodes transmit at each step the rumour to each susceptible neighbour, independently with probability p. After one step, nodes that were infected in the previous step are removed and stop the transmission. Notice that SIR generalizes broadcast, to which it is equivalent when $p = 1$. For consistency with the PUSH-PULL model we will define informed nodes in SIR as the ones either in infected or removed state (i.e. the ones that already received the rumour). One of the most striking results concerning SIR is that the diffusion is governed by reproduction number $R_0 = p\bar{d}$, where $\bar{d}$ is the average degree of the nodes. In random networks with degree distribution of finite variance, $R_0 = 1$ is a critical threshold below which the process extinguishes before infecting a negligible fraction of the nodes. For $R_0 > 1$, instead, the process grows exponentially fast. Such threshold effect is absent in certain random graphs with scale-free (for instance power law) degree distribution [8].

These rumour spreading processes can be regarded as the simplest models for information (or epidemic) diffusion. We can now describe our results and methods more precisely.

Problem definition, methodology and results.

The goal of the paper is to develop a space-efficient predictor that is capable of estimating as precisely as possible the expected number of infected nodes at time t when an epidemic spreads across a real-world social network, where the expectation is taken over all possible runs of the diffusion process under consideration. The input to the predictor is not the network itself but its degree distribution only.

A recent paper by Goel *et al.* [16] shows that the well-known random configuration model can be surprisingly good at predicting qualitatively the shape and evolution of viral processes in Twitter. We explore this insight further by adopting the configuration model as the basis for our predictor. In a nutshell, our approach is, first, to develop a provably exact and space-efficient estimator for the configuration model and, second apply it to real-world networks.

Let us recall the definition of the *configuration model* [37, 16]. We are given m *stubs*, m an even number, partitioned into n *buckets* according to a given *degree sequence* $D := (d_1, d_2, \ldots, d_n)$, where d_i is the number of stubs in bucket i, i.e., the degree of node i in the graph. An undirected (multi)graph is generated by the following random process: a pair of stubs is chosen uniformly at random; let i and j be the buckets these stubs belong to; an edge connecting nodes i and j is inserted in the (multi)graph. The procedure is repeated until there are no more free stubs. Observe that m is even so this procedure ends. The model can be easily generalized to the case of directed (multi)graphs.

In contrast to real social networks, random graphs generated by the configuration model have very few triangles. And yet, as the results in [16] and in this paper show, it can be the basis for accurate predictions of diffusion processes in real-world networks. A very positive feature of the model is its simplicity, which opens the way to a rigorous mathematical analysis. We will make use of this feature by proving that our space-efficient estimator is exact for the configuration model.

By using a random graph model our problem becomes mathematically well-defined: given a degree distribution D, we want to estimate, for each time t, the expected number of nodes that are informed by rumour spreading in a random graph drawn from the configuration model with degree distribution D, starting from a random source.

Notice now that this problem admits a trivial solution via sampling: it is enough to generate "many" random graphs with the given degree distribution and, for each one of them, to simulate rumour spreading (itself a random process) "many" times. If we accumulate the results by computing averages for each value of $t = 0, 1, \ldots$, by the law of large numbers the averages will converge to the expected values. We shall refer to this procedure as the *naive estimator*.

The naive estimator can be easily parallelized, as each sample is independent, but has one clear bottleneck: space. The scale of current social networks would force the use external memory – namely disk storage – unless special hardware is available, resulting in prohibitive running times. Moreover, the naive predictor cannot scale efficiently in distributed computation paradigms like MapReduce, due to the network communication bottleneck, as such paradigms are not suited to run jobs that requires a large shared memory (i.e. the graph).

In this paper, we develop a predictor along the same lines of the naive estimator, but such that, by generating each graph locally "on the fly", it can keep everything in main memory even for very large graphs. In terms of space, our predictor needs only $O(n)$ instead of $O(n + m)$ space which for real-world social graphs is a very significant improvement (m denotes number of edges in the graph). For undirected graphs, further optimization is possible which allows spectacular savings in memory usage. To give a rough idea, storing the entire Facebook network would require 480GB of memory[2], whereas our estimator can be run on less than a gigabyte of memory.

The design of such space efficient algorithms poses interesting algorithmic challenges. In particular the algorithms need to be able to sample a constantly changing distribution in an efficient way. To this end we develop a simple variation of the well-known Walker's method [35] (see Appendix A) which proves to be practically very efficient.

More specifically, our results are as follows. We develop a predictor such that, given as input a degree distribution:

- It gives the correct prediction for our diffusion processes in the configuration model, i.e., given a degree distribution D, for each time t, it correctly computes the average number of nodes that are informed at time t when the rumour starts from a random source of a random graph with degree distribution D;

- It gives good predictions for real-life social networks, provided that the neighbourhood function of the network is not "too far" from that of random networks with the same degree distribution. We also show the converse: when the neighbourhood functions differ the predictor performs poorly.

- It is efficient in terms of space and, in the case of undirected networks, exceptionally efficient.

In their study of the neighbourhood function, Boldi *et al.* show that the neighbourhood function of real networks is an informative statistics able to distinguish social graphs from the web graph [6]. More generally, as we do in this paper, it can be used to tell apart social from non-social graphs. Seen in this light our result is especially intriguing, as it shows that, in the case of gossiping and SIR, social networks behave like an average instance of a random graph.

The rest of the paper is organized as follows. In Section 2 we develop the estimator for the configuration model, and show that it is provably exact. Next section discusses how the estimator can be improved in the undirected case. In Section 4 we describe its performance, which is very good for real-life directed and undirected social networks. In Section 5 we recap and hint at possible interesting research directions stemming from our work. Finally, in Appendix A we describe the technical details of our sampling procedure.

[2]Estimated as $\log(n)$ bits per edge for this snapshot [3].

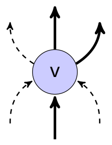


Figure 1: A privy node v during the execution of the algorithm. Solid stubs connect v to (resp. from) other privy nodes. Dashed stubs are free, i.e., represent edges that have not yet been drawn. Node v has $\text{rank}_{\text{in}} = 1$, $\text{rank}_{\text{out}} = 2$, $\text{free}_{\text{in}} = 2$ and $\text{free}_{\text{out}} = 1$.

2. MODEL FOR DIRECTED GRAPHS

In this section we develop a space efficient estimator for the configuration model [37]. In the introduction we have already defined the model for undirected graphs. The directed model that we consider here is essentially identical except that, for each vertex v, we need to specify two quantities, the in and out degree, i.e., the pair $(\deg_{\text{in}}(v), \deg_{\text{out}}(v))$. These pairs must satisfy the obvious requirement that the sum of the in-degrees must be equal to the sum of the out-degrees. If this requirement is satisfied the sequence is called *graphical.*

It is useful to think of each vertex as having $\deg_{\text{in}}(v)$ in-stubs and $\deg_{\text{out}}(v)$ out-stubs. Edges are generated by selecting without replacement a random in-stub and a random out-stub. Doing this can create parallel edges and self-loops but they make for a negligible fraction of the total number of edges when $n \to \infty$ (by a balls-and-bins argument). Hence, the error introduced this way is negligible. Note that the efficient and unbiased generation of simple (as opposed to multi) graphs is an open problem [4].

Let us now switch to the estimator, which will be described for the PUSH process only for lack of space. The discussion for PULL, SIR and broadcast (which is essentially a special case of SIR) is similar, and omitted from this extended abstract.

In what follows by *sample* we mean a run of rumour spreading for a randomly generated graph and a random source. Our goal is to execute each sample efficiently in terms of space. This is achieved by simulating rumour spreading without storing the edges of the graph. The rough idea is to generate edges in a piecemeal fashion when needed, i.e., when an edge is actually used to send the rumour across. We observe that once the edge is used for the transmission of the rumour it can be immediately forgotten, because both endpoints will know the rumour afterwards. Hence, we do not need to store it, but only the fact that its two endpoints now have, respectively, one more out-edge and one more in-edge. The resulting saving, as we shall see, is significant. This approach is similar to the ones used for the analysis of random graph processes [37].

Let us now describe the algorithm that executes samples in a space efficient way. For every vertex u, we shall keep track of whether it is *privy* (it knows the rumour) or *out-of-the-loop* (it does not know the rumour) and the following quantities:

1. $\text{rank}_{\text{in}}(t, v)$, the number of in-neighbours at time t that are privy;

2. $\text{rank}_{\text{out}}(t, v)$, the number of out-neighbours at time t that are privy.

As discussed, the algorithm does not keep track of the edges that are generated – they are effectively forgotten. Instead, for each vertex v, we will remember just how many in and out edges respectively have been generated so far, without knowing which edges exactly they represent. We can think of stubs as being *used* or *free*, and keep track of the following quantities:

1. $\text{free}_{\text{in}}(t, v)$, the number of in-edges that are still undetermined at time t;

2. $\text{free}_{\text{out}}(t, v)$, the number of out-edges that are still undetermined at time t.

In the sequel, we will drop the dependency on time, when it is clear from the context. Clearly, these quantities are somewhat redundant since $\deg_{\text{in}} = \text{rank}_{\text{in}} + \text{free}_{\text{in}}$ and $\deg_{\text{out}} = \text{rank}_{\text{out}} + \text{free}_{\text{out}}$. The algorithm for PUSH actually stores, for each node, only the values $\deg_{\text{out}}$, rank_{out}, and free_{in}. The values $\deg_{\text{in}}$ are only used to initialize the values $\text{free}_{\text{in}} := \deg_{\text{in}}$, the value rank_{in} is not actually used. The other quantities can be simply computed from these, but it is useful to write them explicitly for clarity of the exposition (see Figure 1). Likewise, it is useful for explanatory purposes to think of vertices as having in and out stubs attached to them. They are not actually stored, but they help explaining the algorithm, which we do next.

Its input consists of a graphical sequence D and an integer parameter τ. Its goal is to simulate one run of rumour spreading for τ rounds for a random graph taken from the configuration model given by degree sequence D. It proceeds in a sequence of synchronous rounds $t = 0, 1, 2, \ldots, \tau$ (in practice, as soon as all nodes become privy we can stop the simulation). Initially one node s, the source, is made privy uniformly at random.

For $[0 \le t \le \tau]$: (Refer to Figure 2). All nodes that are privy in the current round are processed in an arbitrary order. Let u be the current privy node to be processed. One of its out-stubs is selected uniformly at random. If the stub is used nothing happens (this corresponds to the fact that the rumour will be sent to another privy node). Otherwise, the selected out-stub of u is free and becomes used. Then, a free in-stub is selected uniformly at random. Let v be the node to which this free in-stub belongs. Next, we increment $\text{rank}_{\text{out}}(u)$, decrement $\text{free}_{\text{in}}(v)$ and mark one of the free in-stubs of v as used. If v is out-of-the-loop, it becomes privy at round $t + 1$ (notice that v remains out-of-the-loop during this round and may be selected several times).

We remark once again that the graphical operations concerning stubs are not necessary for the algorithm but we describe them for clarity of exposition. In particular, we *do not* store the stubs explicitly, but only the quantities $\deg_{\text{out}}$, rank_{out} and free_{in}. Note that the probability that a used out-stub is selected in round t for a given vertex u is $p = \frac{\text{rank}_{\text{out}}(u)}{\deg_{\text{out}}(u)}$.

Time complexity.

We start by analysing the time needed to simulate a single round of the algorithm. We make the standard assumptions [27] that we can sample in constant time for a uniform distribution over $[0, 1]$ and that basic arithmetic operations require $O(1)$ time.

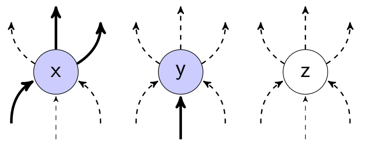


Figure 2: The situation at time t. Privy nodes are dark. Notice that stubs of out-of-the-loop nodes are all free.

The main difficulty in implementing a round is the selection of free in-stubs uniformly at random. This would be easy if we stored the free in-stubs explicitly, which we do not. But since we know, for each vertex, their number, we can equivalently pick a vertex with a probability proportional to the number of its free in-stubs. Performing such random selection efficiently is a non trivial task as the probability distribution of all nodes changes at each edge drawing. This problem is known in literature as dynamic variate sampling [36, 27] and several solutions provides different space-time trade-off. As we provide more details about this operation in Appendix A, here we only summarize the most important results.

The asymptotically optimal Mathias et al. [27] algorithm implements such operations using $O(n)$ space and in $O(1)$ time for each in-stub selection. Wong et al. [36] presented an easy to implement $O(n)$ size data structure that requires $O(\log(n))$ time per sampling. In our experiments we actually use [36] or simple variant of the Walker [35] method that we have developed and which provides almost constant time for $O(n)$ space (more details in Appendix A).

All the remaining updates and random choices (e.g., selection of out-stubs for a given node uniformly at random) can be implemented in $O(1)$ time per node processed. The total number of nodes processed in one step is given by the number of informed nodes. Hence this cost is equal to the number of PUSH messages exchanged. Let $\mu(\tau)$ be the number of messages exchanged by the rumour spreading process in the first τ steps of the process (or up to when all nodes are informed). The complexity of the algorithm is simply $O(\mu(\tau))$. Note that the exact estimation of the message complexity of PUSH and PULL is a challenging problem, for which only partial answers are known [10]. Clearly we have the trivial upper bound of $\mu(\tau) = O(\tau \cdot n)$. On the other hand, note that the simulation of PUSH on a given graph clearly cannot take less than $\mu(\tau)$ time, thus proving the algorithm has the same complexity as simulating the rumour spreading process in a graph. For the PULL and SIR algorithms, not presented in this paper, similar argument shows that our algorithm needs $O(\mu(\tau) + m)$ time to simulate the rumour spreading process for given graph. Nevertheless, the most important advantage of our simulation is the small space requirement, which makes it possible to run each sample in main memory. Simulating samples with external memory would make the task computationally demanding.

Space complexity.

The PUSH algorithm requires for each node in the graph $O(1)$ space to store the necessary information: in and out degrees, in and out ranks and the number of in and out free stubs. This gives $O(n)$ space in total. To select in-stubs uniformly at random, the algorithm employs an additional data structure. As already mentioned with both Mathias et al. [27] and Wong et al. [36] algorithms such data structure requires $O(1)$ space per element of the sampling set, thus the total cost is again $O(n)$. We can conclude that the algorithm requires for directed graphs $O(n)$ space. The same conclusion holds for the PULL and SIR algorithms.

This asymptotic analysis shows the reason why in practice the savings are quite substantial compared to representing the entire graph. We will see later that undirected graphs allow for further optimizations with very interesting results.

Correctness of the algorithm.

In the introduction we discussed pros and cons of the configuration model. One of the crucial properties that we pointed out is its simplicity, which opens the way to rigorous mathematical reasoning. In this section, we take full advantage of this feature by proving the correctness of the algorithm. To proceed formally with the proof we need to introduce the following definitions.

Let D be a fixed graphical sequence. Consider the naive estimator, defined in the introduction: pick a random graph from the configuration model according to D, pick a random source inside this graph, and run rumour spreading for τ rounds. This generates a sequence of privy sets A_i, $i = 0, 1, \ldots, \tau$, where A_i is the set of vertices that are privy at round i. And, let B_i, $i = 0, 1, \ldots, \tau$, the analogous sequence of privy sets generated by the algorithm for the same D.

Theorem 1. For every $i = 0, 1, \ldots, \tau$, the sets A_i and B_i have the same probability distribution.

For lack of space we can only sketch the proof. The details will appear in the full version of the paper. The claim of the theorem holds at the start for the sets A_0 and B_0. They contain only the source, which is selected uniformly at random in both cases.

Effectively the algorithm determines an underlying graph with degree sequence D by drawing its edges when needed by the PUSH process. As the algorithm keeps track, of the number of nodes currently informed in the neighbourhood of each node, we observe that the rumour spreading process is correctly simulated on the underlying graph produced. However, what needs to be proved is that the underlying graph is actually drawn with a probability given by the configuration model. It is actually possible to prove this for a wider class of algorithms. Consider an algorithm where the order of nodes that perform PUSH is arbitrary, i.e., we select an arbitrary privy node with free out-stub and then we choose the destination node by taking u.a.r. a free in-stub. By applying the deferred decision principle on the choices of the in-stubs it is possible to prove that graphs obtained in this way are sampled from the configuration model distribution, which concludes the proof of Theorem 1.

3. MODEL FOR UNDIRECTED GRAPHS

The algorithm described in the previous section can be simplified for the case of undirected graphs. In this section we briefly sketch the main differences between the two algorithms, deferring the complete discussion to the full paper, while pointing out a remarkable feature of the undirected case namely, its economy in terms of space.

In the undirected case, each vertex only has stubs, as opposed to in- and out-stubs. Consequently we only need to keep track of the rank of a node and the number of free edges incident to it. The main difference between the directed and the undirected case is that it is convenient to use the following matrix representation to store the current state of the algorithm. We can encode the current state of all nodes in a matrix of size $O(\Delta^2)$, where cell (i, j) stores the number of nodes with degree i and rank j. Note that only the $i \geq j$ cells are necessary. Because we still need to remember whether the node is privy or out-of-the-loop, we use two such matrices. In other words, we group nodes by degree and rank. To perform the simulation we only need the cardinality of these groups. This is also possible in the directed case, but only if the network is undirected we can achieve significant space optimization.

In particular, this is the case for graphs whose degree distribution follows a power law or heavy tail distribution, as it is typical of real social graphs [30]. In order to give an idea of the kinds of savings involved, let us assume that the degree distribution follows a power law with $\alpha > 1$ (i.e. the fraction $f(k)$ of nodes with degree k is $f(k) = k^{-\alpha} Z(\alpha)^{-1}$, where $Z(\alpha)$ is a normalization constant). Information for nodes of degree $\deg(v) < \gamma$, for a carefully chose threshold γ, is stored in two $\gamma \times \gamma$ matrices. Higher degree nodes are treated individually in a linear array (as in the previous algorithm). Let us now analyse the cost of this representation.

Definition 1. Let $E(\gamma)$ be the cost of the rank data structure with parameter γ. $E(\gamma) = M(\gamma) + H(\gamma)$ where $M(\gamma)$ is the cost of representing the matrices and $H(\gamma)$ is the cost for the information of the high degree nodes.

We have $M(\gamma) = \frac{1}{2}(\gamma)(\gamma + 1) = O(\gamma^2)$, while

$$H(\gamma) = n \sum_{k=\gamma+1}^{\infty} f(k) = O\left(n \int_{k=\gamma+1}^{\infty} x^{-\alpha} dx\right) = O(n\gamma^{1-\alpha}).$$

We can see that the minimum of $E(\gamma)$ is reached for $M(\gamma^*) = H(\gamma^*)$, which gives $\gamma^* = n^{\frac{1}{1+\alpha}}$ and optimal cost $O(n^{\frac{2}{1+\alpha}})$. Since $\alpha > 1$ the cost is always $o(n)$. Moreover, for real networks α is typically in the $(2, 4)$ range (see for instance Table II in [30]). Hence, we can achieve a significant space improvement with respect to storing the entire graph that requires $O(n + m)$ space.

To provide an understanding on the orders of magnitude of the saving, we report in Table 1, a back-of-the-envelope calculation of the number of memory cells needed by the data structure for the best gammas. We need three cells for each entry of the $\gamma^* \times \gamma^*$ matrix and three cells for each high degree node (details are omitted from this paper). The space requirements of each graph are estimated with their number of edges. It is apparent that the space savings with this optimization are very significant.

The data for Facebook (world) is referred to a snapshot of Facebook with 7.2×10^8 active users and 1.4×10^{11} edges, analysed by Backstrom et al. [3]. While we do not have access to the graph, for obvious reasons, the degree distribution was made available by Ugander et al. [33]. Perhaps, this is another confirmation that the access to such statistical information for the research community is easier than the access to the graph.

Note that with [36] algorithm, the additional data structure required to sample edges in such huge graph easily fits in less than 300 MB of memory. This justifies our claim in the introduction that less than 1 GB of memory is sufficient to run our algorithms on such huge graph.

Graph	$\gamma*$	#Cells used	#Edges
AstroPh	54	10,080	393,944
Dblp	91	24,111	6,418,218
Enron	44	7,581	361,622
Facebook (dataset)	84	22,266	1,456,818
Facebook (world)	2,183	1.4×10^7	1.4×10^{11}
Renren	85	27,726	1,410,496

Table 1: Optimal γ values and cost of the compressed representation for real undirected networks (in terms of memory cells used).

The matrix representation can also be used for the directed case, but in this case, as we need $O(\gamma^3)$ space, the savings appear to be modest.

4. EVALUATION

In this section we evaluate the predictive power of our model in real social networks. We have executed the following experiments whose goal is to compare the actual process with the prediction of our estimator. Given a specific real network G we use it to compute the ground truth by means of the naive estimator described in the introduction: for $0 \le t \le \tau$, we compute the expected number of privy nodes at time t by simulating the rumour spreading process on the graph r times. The same approach is used to determine the output of the predictor.

The number r of samples necessary for a certain precision can be determined by standard statistics techniques. Roughly, the variance of the process can be determined empirically through sampling. Knowing the variance, a standard application of the central limit theorem, provides an estimation of the number of samples necessary. Concretely, in the PULL process for instance, a confidence interval of $\pm 0.5\%$ of the nodes in the graph, with probability 0.95, requires for the ground truth approximately between 300 and 5,000 samples depending on the network. On the other hand between 1,000 and 5,000 samples are needed for the predictor. However, to show the most accurate results possible, we have used a number of samples that greatly exceed these numbers, as in most graphs we employed 10,000 samples.

All the diffusion processes analysed can spread the information only to nodes connected to the starting node. For this reason, to be able to measure the accuracy of the estimator in a reliable way, we restrict the analysis to the largest strongly connected component in each graph. Otherwise, depending on the starting node, the number of informed nodes at the end of the process would vary greatly in a way that is not related to the actual rumour spreading process.

Accuracy measures.

The literature on prediction accuracy is vast and several different measures have been defined (see [17] for instance). In this paper, we report results for two well-known measures: the L_2 distance and the Mean Absolute Percentage Error, or MAPE [17].

To introduce formally the accuracy measures employed let us begin with the following definition.

Definition 2. For $t = 0, \dots, \tau$, let $s_t \in R$ be the average number of nodes informed time step t in the rumour spreading process (i.e., the ground truth). Similarly, at the same time let $p_t \in R$ be the average number of nodes informed as predicted by the algorithm.

Note that for consistency of notation, for SIR, we define s_t and p_t as the number of nodes currently in state infected or removed at time t – i.e. the nodes that have received the rumour – in the ground-truth and in the prediction, respectively.

Given these definitions, the well-known L_2 distance of the two series is defined as

$$L_2(S, P) = \sqrt{\sum_{t=1}^{T} |s_t - p_t|^2}.$$

The L_2 distance intuitively measures how far the two curves are from each other in the plane. An L_2 value of 0 shows a perfectly correct prediction (i.e., $s_t = p_t$ for all t) while the maximum error is unbounded. As the two curves s_t and p_t range in $[0, n]$, the L_2 distance clearly depends on the size of the graph and is not well suited to compare directly the prediction error across different graphs (we can expect larger errors on larger graphs). If we consider, however, the two curves scaled down by n, i.e., the fraction of nodes informed in each time step, both $\frac{p_t}{n}$ and $\frac{s_t}{n}$ will range in $[0, 1]$ and the L_2 distance applied to the two scaled vectors will be not dependent on the size of the graph any more. Notice that this measure is exactly equivalent to dividing the L_2 distance of the two (unscaled) curves by n. For this reason, in this section we report the L_2 distance value normalized by the number of nodes.

To complement our analysis we assess the prediction error with another widely-used measure, the Mean Absolute Percentage Error, or MAPE [17], which is defined as follows

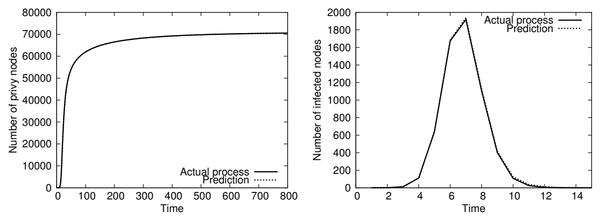
$$\text{MAPE}(S, P) = \frac{1}{T} \sum_{t=1}^{T} \left| \frac{s_t - p_t}{s_t} \right|.$$

The measure intuitively assesses the average percentage of the prediction error with respect to the ground truth and contrary to the L_2 distance it does not require normalization.

While easy to interpret, the use of MAPE (or any other measure based on averages such as RMSPE, GMRAE to name a few [17]) requires a certain care in our context.

Observe that in PUSH and PULL s_t converges to n for $t \to \infty$, but it never reaches this value because the probability that there is an uninformed (out-of-the-loop) node is always non-zero. To see why this is problem, consider the trivial prediction: $p_t = n$ uniformly for all t. For large enough t, s_t is arbitrary close to n (in connected graphs) and thus the MAPE of the trivial prediction converges to 0 as T tends to infinity. Similar considerations apply to SIR where s_t.

In order to overcome this problem we need to compute MAPE for a fixed, and finite, interval $[0, T]$. Intuitively, we want to measure the MAPE only for the part of the curve that is informative – i.e. when the process is still evolving. For this reason we report MAPE for the range $[1, T(\epsilon)]$ where $T(\epsilon)$, in the PUSH and PULL case, is defined as the largest time for which the ground truth has at most $(1-\epsilon)n$ informed nodes. Similarly for SIR, where the ground-truth curve s_t converges for $t \to \infty$ to certain value δ in $[0, n]$ we define $T(\epsilon)$ as the largest time in which at most $(1-\epsilon)\delta$ nodes informed. We estimate δ by executing the process for a large τ number of steps after which a negligible fraction of nodes are still actively infected in expectation. By choosing ϵ, for instance in the interval $[1\%, 5\%]$, we can evaluate the



(a) PUSH (b) SIR

Figure 3: Prediction on directed random graph. Results for PULL are very similar to PUSH

prediction for the interesting part of the trajectory of the process. In Table 3 we report values for MAPE using $\epsilon = 2\%$.

Random graphs.

In this section we report the performance of our predictor for the configuration model. Figure 3 exemplifies the results obtained for directed graphs: confirming Theorem 1 the prediction is essentially perfect (the results in the undirected case are similar). The plots show the outcome of a simulation for which the degree sequence was extracted from a Slashdot snapshot [24]. In both this picture and the following the time represents the number of steps in the process. Both prediction and the actual process have been averaged over 100 random graphs for which we run 100 times the rumour spreading process. We obtained the following L_2/n values for the three processes: 0.0014 for PUSH, 0.0345 for PULL and 0.0095 for SIR. Note that we observed that the PULL and SIR process exhibits an higher variance than PUSH and this is reflected by the higher L_2 norm. Intuitively, this is determined by the large dependence on the degree of the source node in the first steps of the process with respect to the PUSH process. Finally, we need to state that in a series of experiments we have observed that the evolution of the process depends strongly on the degree distribution when n and m are fixed – the results will be included in the full version of the paper.

Real networks.

Let us now discuss the predictions of the model for real networks. We consider both graphs whose nodes and edges represent people and human relationships (henceforth social graphs) and graphs whose nodes do not represent human beings (non-social). Our social graphs can be roughly divided into several groups by their origin:

- *Friendship and trust networks*: to this group belong snapshots of the undirected trust network of Epinions [31]; the undirected friendship networks of Facebook [34] (New Orleans community) and Renren [18]; and the directed friendship networks of Pokec [32], LiveJournal [9] and Slashdot [24];

- *Collaboration networks*: here belong the undirected co-authorship networks in astrophysics (AstroPh [23]) and computer science (DBLP [7]); and the directed communication network of Wikipedia's users (Wiki-Talk [22]);

Graph	$\lvert V \rvert$	$\lvert E \rvert$	d_{eff}	cc	ϕ
AstroPh	17903	393944	5.05	0.63	1.01×10^{-2}
DBLP	805021	6418218	7.35	0.65	4.41×10^{-3}
Epinions	32223	443506	5.34	0.24	4.00×10^{-2}
Enron	33696	361622	4.82	0.51	4.52×10^{-3}
EuAll	34203	151132	4.35	0.26	9.38×10^{-2}
Facebook	59691	1456818	5.14	0.24	2.13×10^{-2}
LiveJournal	3828682	65349587	6.70	0.31	2.40×10^{-4}
Pokec	1304537	29183655	5.85	0.12	3.45×10^{-2}
Renren	33294	1410496	4.26	0.20	3.22×10^{-2}
Slashdot	71307	841201	4.77	0.07	2.33×10^{-2}
Wiki-Talk	111881	1477893	3.97	0.18	1.60×10^{-1}
Amazon	241761	1131217	25.5	0.40	4.27×10^{-3}
BerkStan	334857	4523232	16.21	0.64	1.55×10^{-4}
Google	434818	3419124	13.99	0.64	4.56×10^{-4}
NotreDame	53968	296228	14.27	0.55	1.29×10^{-4}
Stanford	150532	1576314	16.31	0.64	1.47×10^{-4}
RoadNet	1957027	5520776	503.44	0.05	2.04×10^{-3}

Table 2: Properties of the graphs analysed. For all graphs we consider only the biggest (strongly) connected component. Light and dark grey cells represents, respectively, social and non-social networks.

- *Email networks*: undirected mail exchange networks of Enron [24] and of a research institution (EuAll [23]);

For what concerns the non-social graphs we evaluated:

- *Web networks*: here we have directed snapshots of the Notre Dame university (NotreDame [1]), Stanford university (Stanford [24]) and Google [24] websites and a crawl of the pages of Stanford and Berkley university (BerkStan [24]).

- *Product networks*: here belong a directed product co-purchasing network form Amazon websites (Amazon [21]).

- *Road networks*: here we consider a directed graph representing the connections of the roads of California (RoadNet [24]).

All graphs analysed are available online or on request. For each graph, we considered only the largest (strongly) connected component. Table 2 reports some statistics for these graphs: d_{eff}, cc and ϕ indicate, respectively, an estimation of the 90-th percentile effective diameter [23], the clustering coefficient and conductance [24], obtained with the SNAP graph library.[3] The conductance was estimated with the method of Andersen et al. [2], removing orientations in the case of directed networks.

In what follows, for lack of space, we report results that exemplify our main conclusions. The full data sets are available on request and will be publicly distributed with our code and appear in the full paper.

The main qualitative conclusion that our experiments warrant is that the accuracy of the prediction of the estimator follows a very different trend in the two main categories of network analysed. For all instances of what we dubbed social graphs the prediction matches qualitatively well the evolution of the process in the actual graphs. Remarkably, the prediction is especially accurate for friendship and trust networks (Epinions, Facebook, LiveJournal, Renren, Slashdot) – i.e., ones that could be considered to be more "genuine" social networks (See Figure 6 for some examples). For

[3]Available at `snap.stanford.edu/snap`, the same website provided many of the graphs used.

	PUSH		SIR		Neighbourhood F.	
Graph	L_2/n	MAPE	L_2/n	MAPE	L_2/n	MAPE
AstroPh	0.549	0.208	0.647	0.470	0.508	0.222
DBLP	1.32	0.614	0.698	0.521	0.564	0.263
Epinions	0.536	0.032	0.256	0.286	0.314	0.100
Enron	2.74	0.062	0.327	0.316	0.212	0.056
EuAll	2.05	0.093	0.576	0.368	0.342	0.069
Facebook	0.476	0.105	0.448	0.316	0.428	0.182
LiveJournal	0.688	0.216	0.467	0.423	0.632	0.455
Pokec	0.356	0.005	0.631	0.137	0.359	0.188
Renren	0.490	0.042	0.235	0.147	0.249	0.099
Slashdot	0.378	0.019	0.048	0.044	0.118	0.035
WikiTalk	1.17	0.029	0.061	0.070	0.094	0.024
Amazon	5.24	15.9	0.023	67.3	2.87	4.70
BerkStan	3.61	0.187	0.755	51.3	1.89	0.104
Google	4.05	0.648	2.15	225.4	1.78	2.67
NotreDame	4.79	0.267	2.03	19.3	1.41	0.338
Stanford	4.07	0.162	0.401	11.3	1.70	0.200
RoadNet	23.9	67.9	2.623	24412.2	14.8	48.9

Table 3: Accuracy of the algorithms presented (MAPE is calculated for $\epsilon = \%2$). We report the results of undirected algorithm for undirected graph and directed algorithm for directed graphs. Light and dark grey cells represents, respectively, social and non-social networks. Notice the much higher prediction errors for non-social graphs. Results for PULL are very similar to PUSH and omitted for lack of space.

other social graph classes, the prediction error is comparably larger, but still is qualitatively close to the ground-truth. In particular for some instances of collaboration networks such as DBLP and the mail networks (EuAll, Enron) it works rather well.

In sharp contrast, the prediction in non-social graphs (web graphs, product networks and road networks) is very inaccurate. Figure 6 exemplifies these findings in the case of PUSH. Similar results are obtained for PULL and SIR (omitted from this extended abstract). In all the reported experiments involving SIR we report the experiments where the parameter p was set such that the reproduction number $R_0 = p\bar{d}$ is 1. Similar results are obtained for other parameter settings. Table 3 reports the MAPE and L_2 errors for all our networks.

For each network we have averaged over 10,000 samples, except for LiveJournal for which 1,000 samples were used due to running time constraints. The number of steps simulated was $\tau = 3{,}000$, after which the all processes simulated change by only a negligible fraction in expectation.

Discussion.

The sharp distinction in the prediction accuracy between social and non-social graphs is intriguing as it suggests that the two classes of graphs behave differently with respect to the rumour spreading process. The former shows a close similarity to the evolution of the rumour spreading processes in the configuration model graphs while the latter does not. Is there some specific property of these type of networks that makes our prediction accurate?

Several properties are known to distinguish friendship graphs from non-social networks: diameter (look at Table 2), assortativity [28][4], and compressibility [7], some of which has been already reported to influence epidemic processes [5].

But the key to interpret the results is the neighbourhood function [6]. The neighbourhood function $f_G(t)$ of a graph $G = (V, E)$, for $t \in 1, \dots n$, is a defined as the number of ordered node pairs (u, v) such that v can be reached from u by a directed path of length at most t. This function, and in particular its so-called index of dispersion [6], has been shown to be able to tell apart social graphs from the web graph. Hence it is a good candidate to explain the different behaviour observed in our experiments.

Moreover, consider the SIR process with probability of transmission 1 (i.e., the broadcast). It is possible to prove that the expected number s_t of nodes informed at time t, for a u.a.r. chosen source node, is linearly related to the neighbourhood function $f_G(t)$ of the graph, more precisely $s_t = \frac{f_G(t)}{n}$.

Hence, if we observe that a real network G has a neighbourhood function that closely resemble that of an average instance of configuration model graphs with the same degree distribution, this would imply that our prediction for the broadcast process in such network is accurate. The opposite is also true, the broadcast process in a graph with a neighbourhood function very different (in a way that has to be formally specified) with respect to the configuration model graphs will be predicted inaccurately by our model. Notice that this mathematical implication holds only for the broadcast model. However, in any of the process evaluated (PUSH, PULL and SIR with arbitrary probability) it is still possible to notice that the expected number of nodes informed s_t, starting from a random source, is upper-bounded by the neighbourhood function, more precisely $s_t \leq \frac{f_G(t)}{n}$.

It is hence possible to expect a close relationship between the similarity of the neighbourhood function of a graph with that of configuration model graphs and the precision of our prediction method in such graph. We tested this hypothesis with the following experiment. For each real graph G, we computed the neighbourhood function $f_G(t)$ of the graph and the average neighbourhood function over random configuration model graphs with the same degree distribution as G. Then we measured the distance between the two functions (for instance with the L_2 norm) and compared this distance with our prediction error.

The main message is exemplified by Figure 7. On the left hand side we see what happens for the Slashdot network: its neighbourhood function is very close to that of a random network and the prediction of the average number of infected nodes (in this case by SIR) is very accurate. In contrast, on the right, we see that when the neighbourhood functions are far apart the prediction is off. Notice that for the random networks we use the average of the neighbourhood function over 10 instances of the configuration model. In both real and random graphs we compute the neighbourhood function using the method of Boldi et al. [6].

Table 3 reports the full data of our experiments. The results show a clear and strong correlation between the distance of the two functions and the approximation error. The last two columns in the table report the distance of the two neighbourhood functions for our real networks. Notice that the distance is much higher in non-social graph than in social graph (for instance L_2/n is at most ~ 0.6 in the former and at least ~ 1.4 in the latter).

To further validate the hypothesis, for the 17 real graphs in our dataset, we tested the Pearson correlation coefficient between the L_2/n distance of $f_G(t)$ and $c_G(t)$ in a graph G and the MAPE error of the prediction. This coefficient is 0.98, 0.98, 0.97 for PUSH, PULL and SIR, respectively. We

[4]For an illuminating discussion on the correct way to define and use this concept see [25].

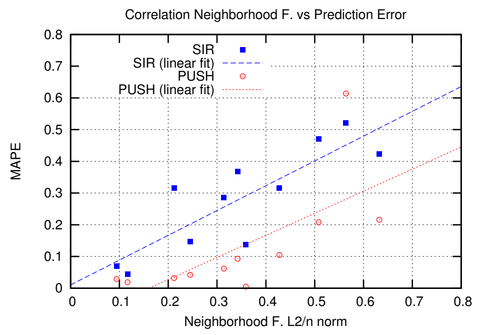


Figure 4: Correlation between prediction error and L_2/n distance between the graph and the average configuration model neighbourhood function. Results for PULL are very similar to PUSH.

Graph	Ground-Truth	Estimated	Error
AstroPh	4.45	5.53	0.24
DBLP	5.74	6.90	0.20
Enron	2.53	2.64	0.04
Epinions	3.18	3.38	0.06
EuAll	2.61	2.06	0.21
Facebook	5.13	5.60	0.09
LiveJournal	6.17	6.11	0.01
Pokec	10.03	13.57	0.35
Renren	5.12	5.55	0.08
Slashdot	2.83	2.69	0.05
WikiTalk	1.79	1.69	0.06

Table 4: Accuracy of the prediction of the average structural virality of SIR in our social graphs. Notice the small relative with the ground-truth.

checked that the correlation remains strong even considering only graph in the single category (social or non-social). For instance in the case of SIR, the coefficient is 0.84 and 0.99 in social and non-social graphs, respectively. Figure 4 shows the correlation for social graphs only. Notice that all these Pearson coefficients are significant with p-value lower than 0.5%.

Finally, we also checked that these observations are not explained by simpler graph statistics such the clustering coefficient or the diameter. We did not found any statistically significant correlation between MAPE error and conductance or clustering coefficient. On the other hand, the effective diameter has a strong positive correlation with the MAPE over the entire dataset but this correlation becomes non-significant (p-value $> 0.5\%$ in SIR) when looking at single network classes (i.e., only for social or non-social networks).

This result can suggest that, despite its simplicity, the configuration model qualitatively matches real social graphs in terms of their neighbourhood function and hence it justifies the good prediction of our model in such networks and conversely the poor results for non-social graphs.

Structural virality.

We run an additional experiment to verify the predictive power of our model in real networks. In their study Goel *et al.* [16] introduce a scalar quantity called structural virality and argue that to some extent it captures the shape of a diffusion process. Consider a *diffusion tree* T where an edge (u, v) is present if u informed v. The virality of the tree T is defined as

$$v(T) = \frac{1}{n(n-1)} \sum_{i,j} d_{ij},$$

where d_{ij} is the distance in the tree of the nodes i and j. They find this measure to be close in the case of real twitter propagations and SIR in the configuration model.

This measure can be computed efficiently on-the-fly in our framework without affecting the computational complexity of our algorithms. In particular we run the following experiment for the SIR model on our social networks. The results show that the average virality predicted by our model is close to the one observed in the ground-truth process. In our social graphs (see Table 4) the relative error with the ground-truth in the estimation of the virality is between $\sim 1\%$ in Livejournal to $\sim 35\%$ in Pokec. This shows that the model is able to predict qualitatively well not only the average number of informed nodes at each time step but also the viral structure of the diffusion.

Real rumours.

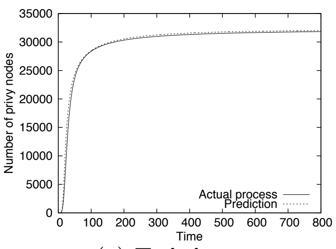
Finally, we have also analysed some real instances of rumour spreading in Twitter. We have collected the tweets for selected hashtags that are related to instances of viral information diffusions.[5] For each of the rumours we tried to fit the prediction of the SIR model on a configuration model graph with power law distribution to the real data with a linear transformation. Figure 5 shows the fraction of nodes informed in the actual twitter process (with respect to the peak value) and in the fitted SIR one. Similar to the experiments in [16] we run SIR on a configuration model graph with power law distribution (we used the Snapshot degree distribution as a reference). In all instances we used the same SIR estimation obtained with parameter $R_0 = 1$ and optimized the linear transformation for the lowest L_2 error.

Notice that the parameters of the linear transformation that best fit the process varies from rumour to rumour (as the number of informed nodes and the speed of the diffusion can be vastly different) so this result is clearly not to be intended that a single instance of SIR can predict *all* instances of rumour diffusions in Twitter. The takeaway observation however is that the SIR model on configuration model graph qualitatively matches the evolution of some viral rumours in real networks, as previously observed by S. Goel et al. [16], thus further validating our approach.

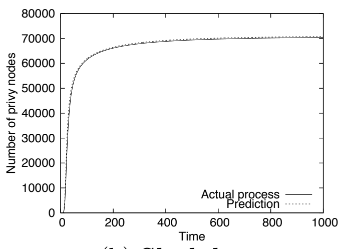
5. CONCLUSIONS

In this paper we have developed a space efficient estimator that is able to predict with good accuracy the average growth of rumour spreading in a given social network. The results are especially accurate for friendship and trust networks. Remarkably, in our opinion, the only information that the predictor needs is the degree distribution of the network. The estimator is based on the configuration model, for which it was formally proven to be correct. We would like to remark that we will make the code and our results available online when the paper is published. This could facilitate to explore the following further avenues of research that our work points to.

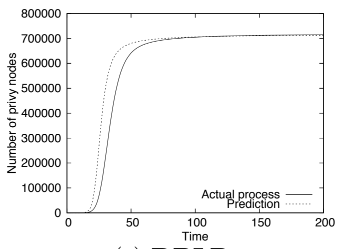
[5]For instance tweets related to important world-wide events, e.g., tweets containing hashtag #*conclave* in the latest Pope election.



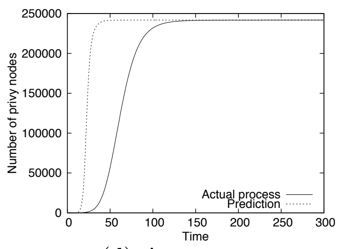
(a) **Epinions**



(b) **Slashdot**

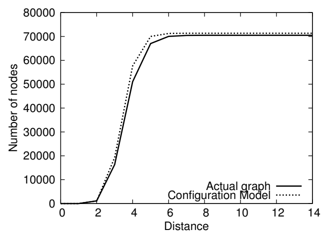


(c) **DBLP**

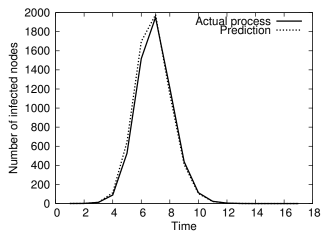


(d) **Amazon**

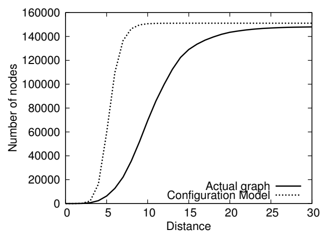
Figure 6: Prediction for the PUSH model. Epinions and Slashdot exemplify the very good prediction obtained for friendship and trust networks; the performance for DBLP is typical of email networks and the worst instances of collaboration networks; finally, Amazon shows the typical result for non social networks.



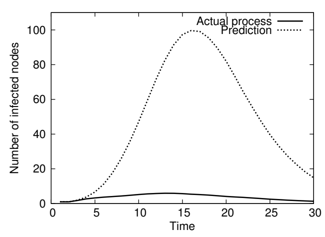
(a) Slashdot - Neighbourhood F.



(b) Slashdot - Prediction

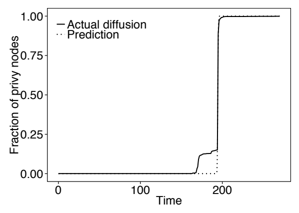


(c) Stanford - Neighbourhood F.

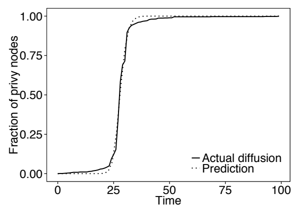


(d) Stanford - Prediction

Figure 7: Neighbourhood function and prediction for SIR in social and non-social graphs. Notice how the social graph Slashdot shows both a Neighbourhood function close to the average one in configuration model graphs and good prediction. The web graph Stanford shows a substantially different neighbourhood function with respect to configuration model and a poor prediction.



(a) #conclave



(b) #EuropeLeague

Figure 5: Fitting of SIR model prediction with selected hashtags diffusion from Twitter in March 2013. The data contains tweets with keywords "#conclave *signaling*" (2108 tweets) and "#EuropeLeague *Steaua*" (2351 tweets).

The first question is whether it is possible to come up with much more compact predictors, for instance based on systems of differential equations [29]. Another interesting direction is to consider a similar approach but for other diffusive processes. For which processes can we predict their growth based only on limited information? In addition, it would be interesting to apply similar techniques to the influence maximization [19] in networks for which the topology is unavailable. Finally, it could be worthwhile trying to develop similar predictors based on more nuanced graph models, that would allow for more accurate predictions and to extend successfully the approach to larger classes of real networks.

Acknowledgements

We thank Flavio Chierichetti and Silvio Lattanzi for many insightful comments, and Nelly Litvak for some useful pointers to the literature.

APPENDIX

A. SAMPLING

The most crucial and time consuming operation in our algorithms is drawing an element form a set with a probability proportional to some dynamic integer value. This problem is known in literature as dynamic variate sampling and several solutions are known [36, 27]. More precisely, we are interested in a data structure that, for a sequence of N non-negative weights summing up to M, provides two operations: *get* (that retrieves an element) and *update* (that modifies its associated weight)[6]. Ideally, we would like to have a data structures that require $O(N)$ space while providing $O(1)$ amortized time for *get* and *update*.

A well-known data structure based on binary trees [36] provides $O(N)$ space and $O(\log(N))$ time for both *get* and *update*. While slower than optimal, this data structure, has the clear advantage of having very low hidden multiplicative constants, making possible to retain almost all the memory saving described in the previous sections. Nevertheless, there are solutions that use slightly more space but support both operations in $o(\log N)$ time.

[6]In our context, $M = m$ is the number of edges m and N is the number of equivalences classes of nodes ($N = n$ for directed graphs, $N = o(n)$ for undirected ones).

Dynamic sampling.

For the case in which weights are fixed, the well known Walker's alias method [35] provides an elegant and optimal solution. The algorithm requires an $O(N)$-time preprocessing step that has to repeated for each update operation. However, for the case of monotone decreasing weights, which applies to directed graph algorithms, we can define a simple and efficient variants of the algorithm based on rejection sampling. Each element is associated with an additional *reject* probability, given by the amount weight reduced by the previous extractions. Every time an element i is drawn, we reject it with $reject_i$ probability and repeat the sampling.

It is easy to show that this reproduces the right distribution, and that the extraction per *get* operation can be constrained to $O(1)$ in expectation. Suppose, in fact, that overall the *reject* probability is less than $\frac{1}{2}$, then each *get* operation would require less than 2 samples in expectation. We can then repeat the $O(N)$ time preprocessing every time the total weight has been decreased by half. The total cost for $O(M)$ *get* and *update*s is then $O(M)$ plus $N \log(M)$ for the initializations. In our algorithms this gives $O(1+\frac{\log(m)}{d})$ amortized cost per *get*, where d is the average degree. This is practically $O(1)$ because d is in general larger or comparable to $\log(m)$.

This algorithm while easy to implement and very efficient, cannot be applied to the case where the matrix data structure is used. In such case, weights can increase as well – nodes can be moved to a new class. However, for this case, Matias et. al [27] algorithm provide a constant time solution which uses space $O(N)$. This general purpose data structure can be applied to all our algorithms proving that $O(1)$ time and $O(N)$ is achievable as stated in the previous sections. Although optimal in the asymptotic sense, this data structure is rather complex to implement. While we have not conducted a thorough experimental evaluation, we found that the simpler data structures discussed previously behave quite well in practice, and are easy to implement.

B. REFERENCES

[1] R. Albert, H. Jeong, and A. L. Barabasi. The diameter of the world wide web. Nature, 1999.

[2] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In FOCS, 2006.

[3] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. In WebSci, 2012.

[4] M. Bayati, J. Kim, and A. Saberi. A sequential algorithm for generating random graphs. Algorithmica, 2010.

[5] M. Boguá, R. Pastor-Satorras, and A. Vespignani. Epidemic spreading in complex networks with degree correlations. Statistical mechanics of complex networks, 2003.

[6] P. Boldi, M. Rosa, and S. Vigna. HyperANF: Approximating the neighbourhood function of very large graphs on a budget. In WWW, 2011.

[7] P. Boldi and S. Vigna. The WebGraph framework I: Compression techniques. In WWW, 2004.

[8] S. Chatterjee, R. Durrett, et al. Contact processes on random graphs with power law degree distributions have critical value 0. The Annals of Probability, 2009.

[9] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On compressing social networks. In KDD, 2009.

[10] F. Chierichetti, S. Lattanzi, and A. Panconesi. Almost tight bounds for rumour spreading with conductance. In Proc. 42nd STOC, 2010.

[11] F. Chierichetti, S. Lattanzi, and A. Panconesi. Rumour spreading and graph conductance. In SODA, 2010.

[12] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In PODC, 1987.

[13] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Randomized broadcast in networks. In Algorithms. Springer, 1990.

[14] N. Fountoulakis, K. Panagiotou, and T. Sauerwald. Ultra-fast rumor spreading in social networks. In SODA, 2012.

[15] G. Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In T. Schwentick and C. Dürr, editors, STACS, 2011.

[16] S. Goel, A. Anderson, J. Hofman, and D. Watts. The structural virality of online diffusion. Under review `5harad.com/papers/twiral.pdf`, 2014.

[17] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. Int. J. Forecasting, 2006.

[18] J. Jiang, C. Wilson, X. Wang, P. Huang, W. Sha, Y. Dai, and B. Y. Zhao. Understanding latent interactions in online social networks. In IMC, 2010.

[19] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003.

[20] W. O. Kermack and A. G. McKendrick. Contributions to the mathematical theory of epidemics. II. The problem of endemicity. Proc. of the Royal society of London. Series A, 1932.

[21] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. ACM Trans. Web, 2007.

[22] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In CHI, 2010.

[23] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. ACM Trans. KDD, 2007.

[24] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. J. Int. Math., 2009.

[25] N. Litvak and R. van der Hofstad. Uncovering disassortativity in large scale-free networks. Phys. Review E, 2013.

[26] D. Maki and M. Thompson. Mathematical models and applications: with emphasis on the social, life, and management sciences. Prentice-Hall, 1973.

[27] Y. Matias, J. S. Vitter, and W.-C. Ni. Dynamic generation of discrete random variates. In SODA, 1993.

[28] M. E. J. Newman. Mixing patterns in networks. Phys. Rev. E, 2003.

[29] B. Pittel. On spreading a rumor. SIAM J. Appl. Math., 1987.

[30] A. Reka and Barabási. Statistical mechanics of complex networks. Rev. Mod. Phys., 2002.

[31] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. International Semantic Web Conference, 2003.

[32] L. Takac and M. Zabovsky. Data analysis in public social networks. In Int. Scientific Conf. Present Day Trends of Innovations, 2012.

[33] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. CoRR, 2011.

[34] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In WOSN, 2009.

[35] A. J. Walker. An efficient method for generating discrete random variables with general distributions. ACM Trans. Math. Softw., 1977.

[36] C. Wong and M. Easton. An efficient method for weighted sampling without replacement. SIAM J. on Computing, 1980.

[37] N. Wormald. Models of random regular graphs. In Surveys in Combinatorics. Cambridge U. Press, 1999.