History Sensitive Cascade Model

Maksim Tsikhanovich and Georgi Smilyanov

July 31, 2009

Abstract

We introduce the history sensitive cascade model, a model of information cascade through a network over time. We consider the ‘activation’ problem of finding the probability of a particular node receiving information given that some nodes are initially informed. Finally, we prove that selecting a set of k nodes with greatest expected influence is NP-hard, and use results from submodular functions to provide a greedy approximation algorithm with a $1 - 1/e - \epsilon$ lower bound, where ϵ depends polynomially on the precision of the solution to the ‘activation’ problem. We perform experiments in order to compare the greedy algorithm to three other approximation algorithms.

1 Introduction

Our research project focuses on network diffusion. This is the process by which some nodes in a network influence other, neighboring, nodes. For example, we start with a network where some of the nodes are in *active* state and the rest are *inactive*. Depending on the different diffusion models (see Related Works), the nodes that are already active can influence their neighbors to become active as well.

Many real-world phenomena can be seen as network diffusion. Examples include the spread of infectious diseases or the spread of new ideas and technologies [Gla00] [Sur04]. Classic instance of diffusion is found in marketing and experimental economics [Wor08] [KKT03] [KKT05]. Consider a company that introduces a new product to the market. It might try to identify the consumers with the largest willingness to pay and sell it to them. But this might not guarantee profit maximization in the long run. Suppose instead that the company finds the people with most influence over some group (e.g. friends, family) and offers the product to them. Building on this plausible scenario, Domingos and Richardson [DR01] introduce the notion

of “network value” of a consumer and define it as the influence the consumer has on others in her social network:

A customer whose intrinsic value is lower than the cost of marketing may in fact be worth marketing to when her network value is considered. Conversely, marketing to a profitable customer may be redundant if network effects already make her very likely to buy.

A diffusion model is commonly used to answer the following question:

- What is the best set of nodes to activate initially so that the number of active nodes in the end of the diffusion process is maximized?

It is NP-Hard in all models we mention in Related Works. However, approximation algorithms for obtaining reasonably good solutions exist.

The main difference between the existing models and History Sensitive Cascade Model we propose is that the latter allows nodes to try to activate their neighbors more than just once. This is a better assumption as in the reality people interact with each other multiple times over some time period.

In this paper we give a formal description of the History Sensitive Cascade Model (HSCM) and compare it to other commonly used diffusion models. We also present algorithms for inference in HSCM and analyze them both theoretically and empirically.

2 Related Works

So far two models of network diffusion have been extensively studied [KKT03], [KKT05], [MR07]. [KKT03] gives a proof of their equivalence.

In the *Linear Threshold Model* a node becomes active when a certain (weighted) fraction of its neighbors is active [KKT03] [Wor08] [Zha09]. In the next time step all nodes that have been previously activated remain active and can also influence their own neighbors. The property that once active, nodes cannot be deactivated is called *progressiveness*. Network effects in economics can be naturally modeled using LTM. For example, someone decides to switch to a new operating system or a new cell phone service provider once the latter gains a sufficient number of users.

In the *Independent Cascade Model* an active node has a one-time chance to activate a neighboring node and succeeds with certain probability [KKT03] [Wor08] [Zha09]. If the neighbour is activated, it remains such for the next

time steps. But the active node cannot make any more attempts to activate the neighbor if the first activation has failed. ICM is good for modeling situations where decisions are taken from the first time and cannot be revised even after more information is added.

The *History Sensitive Cascade Model* (HSCM) we introduce is essentially an extension of the Independent Cascade Model. The main difference between the two is that HSCM allows nodes to try to activate their neighbors more than just once [Zha09].

3 Problem Definitions

There are several interrelated problems we are trying to solve, all of them are related to the History Sensitive Cascade Model, which we now introduce formally.

Definition 1. History Sensitive Cascade Model The HSCM model is represented by a directed and weighted graph $G = (V, E)$; V represents the finite set of vertices in the graph, and E represents the finite set of edges in the graph. Furthermore, there exists a function $A : V \rightarrow \{0, 1\}$, where for each vertex $v \in V$, $A(v) = \{1 \text{ if } v \text{ is active, } 0 \text{ if } v \text{ is inactive}\}$. The HSCM model takes on states, where each state is defined as the set of vertices that are active, and inactive, as a result there are $2^{|V|}$ distinct states. The HSCM model transitions between states over a time step. We say the model is on the t^{th} time step if t time steps have passed. Over any given time step, any active vertex $v1$ can activate any inactive vertex $v2$ with probability $w(ev1v2)$ if $ev1v2 \in E$, where $w : E \rightarrow [0, 1]$ is the function that returns the weight of a given edge, and $ev1v2$ denotes the edge from $v1$ to $v2$. Furthermore, an active vertex can't be deactivated, and therefore HSCM is *progressive*.

3.1 Vertex Activation Problem

The first problem we are trying to solve is called the Vertex Activation Problem (VAP), and we take its definition from [Zha09], where it is defined as.

Definition 2. Vertex Activation Problem Given an instance of an HSCM, and a time step t , what is the probability that a certain v is active on the t^{th} time step?

What we are working on a problem about the Vertex Activation Problem, is to try and determine whether it falls in the class P, or NP, assuming $P \neq NP$. Kempe, et al declared it an open and interesting problem in [KKT03], and by the publication of [KKT05], the authors were still unable to solve this problem. So far we have not been successful either; we have chosen to assume that the Vertex Activation Problem is in NP, and worked on bounding the approximation problem, which can be stated as follows.

Definition 3. VAP Approximation Problem Let $VAP : G \times v \times V \times \mathbb{N} \rightarrow [0, 1]$ be $VAP(G, v, A, t)$ is the probability of vertex v being active in the HSCM instance represented by graph G , with initially active vertices A and time step t . Find a function $Z : G \times v \times V \times \mathbb{N} \rightarrow [0, 1]$ such that Z runs in time polynomial to all its inputs, and either a) $\alpha Z(G, v, A, t) \geq VAP(G, v, A, t)$ for some lower bound $\alpha \in (0, 1)$ or b) $P[Z(G, v, A, t) = VAP(G, v, A, t)] \geq \beta$ where P is the probability function, and $\beta \in (0, 1]$.

3.2 Influence Maximization Problem

Once we have the solution to the Vertex Activation Problem, or an approximation, we use the solution to solve a problem about a set V of vertices, particularly, we ask what is the expected amount of the vertices in V activated by the t^{th} time step if some set of vertices A will be chosen to be active initially. We call this amount of vertices the *influence* of the initially active set A for k time steps.

Definition 4. Influence Function We define the influence function $\sigma : V \times \mathbb{N} \rightarrow \mathbb{R}$ for a particular instance of the HSCM model represented by graph G and for a time step t as $\sigma(A, t) = \sum_{v \in V} VAP(G, v, A, t)$.

This use of the solution to the Vertex Activation Problem lends itself to be integrated as part of a solution for an optimization problem. The optimization problem is the Influence Maximization Problem, which is stated as follows.

Definition 5. Influence Maximization Problem Given a HSCM, whose graph is $G = (V, E)$, and a time step t , which subset U of V , should be selected such that $\sigma(U, t)$ is maximum?

In the next section we show why this is NP-hard; therefore, as with the Vertex Activation Problem, we focus on solving an approximation problem. The approximation problem related with the Influence Maximization Problem is stated as follows,

Definition 6. Given a HSCM represented by the graph $G = (V, E)$, which subset U of V should be selected such that $\sigma(U, t) \geq \alpha \sigma(U^*, t)$, where U^* denotes the optimal solution for some $\alpha \in (0, 1]$?

Once again the goal is to find a polynomial-time approximation algorithm. α represents a lower bound to the performance algorithm, and the goal is to reach the largest possible α . Alternatively as with the Vertex Activation Approximation problem (see Definition 3) we may ask what is the probability of our algorithm returning the optimal solution, or even more likely we will end up saying what is the probability that α takes on a particular value.

4 Solutions

4.1 Vertex Activation Problem

Following the work of [Zha09],[Fos08] we use a state-transition matrix, which we use as part of a Markov chain in order to solve *VAP* exactly. Entry i, j in the state transition matrix is the probability of the system transitioning from state i to state j , where each state is determined by which vertices are active in it; Therefore there are $2^{|V|}$ total entries in the matrix. Each can be calculated in polynomial time: by examining which vertices it's possible to activate from state i and determining the possibility that only those in j are activated, and no others. Once this matrix is calculated, we use a result from Markov chains: we multiply it t times to simulate the system for t steps, and multiply it by a vector which determines the initial state. Finally, we sum the probability of the matrix being in a state where the vertex we're interested in is active, and return this sum as the answer to *VAP*.

Since the matrix is of dimension $O(2^{|V|})$ and we can multiply matrices of size n in $O(3^n)$, it follows that we can solve *VAP* exactly in $O(2^{3|V|})$ time. This result is open to two different interpretations. First, since this algorithm runs in exponential time, and no time is spent doing useless work, one may intuitively believe that the Vertex Activation Problem is NP-hard. On the other hand it is not surprising that this algorithm runs in exponential time, because a model with a transition matrix is reducible to 3SAT, so maybe the exponential runtime is caused by the particular algorithm we use, and is not an inherent property of the problem.

4.2 Vertex Activation Approximation Problem

We approximate vertex activation by sampling. We define the experiment we are working with as follows,

Definition 7. Given a HSCM instance represented by the graph G , a set of initially active nodes A , and a number of time steps, t , execute the history sensitive cascade, described in Algorithm 1.

The possible outcomes of the experiment define the sample space, in our case this is $\{T : T \in \mathcal{P}(V)\}$, where each element refers to the set of vertices that are active. We define the random variable Xi as

$$Xi = \begin{cases} 1 & \text{if } vi \in Tc \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Where Tc refers to the current outcome of the experiment. Let $VAP(G, vi, A, k)$ be the function that solves the Vertex Activation Problem (see Definition 2), and thus it returns the probability that for an HSCM instance represented by the graph G , a initially active set A and a time step k , that the vertex vi is active. This is the function we are interested in approximating. Notice that $VAP(G, vi, A, k) = P(Xi = 1)$ for the experiment performed over G, A, k . Also notice that the expectation of Xi ,

$$E[Xi] = \mu = 1P(Xi = 1) + 0P(Xi = 0)$$

is equivalent to VAP .

Phrasing VAP in terms of determining the mean of a distribution based on samples lets us use standard statistical results to bound how well or poorly our sampling will approximate VAP . We follow the methods suggested by Law in [Law07]. Suppose we have a confidence level, $1 - \alpha$, and a interval half length of β , and we want to know how many samples are needed in order for

$$1 - \alpha \approx P(\bar{Xi} - \beta \leq \mu \leq \bar{Xi} + \beta) \quad (2)$$

to be true, where $\bar{Xi}$ is defined as the sample mean, defined as

$$\bar{Xi} = \frac{\sum_{j=1}^n Xi_j}{n} \quad (3)$$

where Xi_j refers to the value of Xi on the j^{th} of n experiments. The number of samples necessary to achieve this bound is given by

$$N(\alpha, \beta) = \min \left\{ k : tk - 1, 1 - \alpha/2 \sqrt{\frac{\bar{S}^2}{k}} \leq \beta \right\} \quad (4)$$

where $tk - 1, 1 - \alpha/2$ is the upper $1 - \alpha/2$ critical point for the t distribution with $k - 1$ degrees of freedom. $\bar{S}^2$ denotes the sample variance,

$$\bar{S}^2 = \frac{\sum_{j=1}^n (X_{ij} - \bar{X}_i)^2}{n - 1} \quad (5)$$

This result follows from the Central Limit Theorem [Chu74], and it is the case that as $n \rightarrow \infty$, then a) the distribution of the X_i 's becomes normal and b) $\bar{S}^2 \rightarrow \text{Var}(X_i)$. Then Equation 4 also provides an exact $100(1 - \alpha)$ percent bound, instead of an approximate one.

We use an algorithm that computes I samples of X_i in $O(It|V||E|)$ time, where t is the number of steps the model should be able to go through. We present the algorithm to compute one sample.

Algorithm 1 HSCM Sampling Algorithm for Cascade of Length t

Require: $t > 1$, HSCM instance represented by G

```

for  $i = 1$  to  $t$  do
  for all  $v \in V$  do
    if  $A(v) = 1$  then
      for all  $u \in V: evu \in E$  do
         $r \leftarrow$  a random number in  $[0, 1]$ 
        if  $r < w(evu)$  then
           $A(u) \leftarrow 1$ 
        end if
      end for
    end if
  end for
end for

```

This algorithm simply simulates the HSCM process. In order to achieve a some bound related to β with $100(1 - \alpha)$ percent certainty, we keep executing the sampling algorithm while the number of iterations we have done is less than $N(\alpha, \beta)$. We use the mean and standard deviation of the sample we have so far for evaluating $N(\alpha, \beta)$.

4.3 Influence Maximization Approximation Problem

First we prove that the exact version of the problem (see Definition 5) is NP-hard. Then we provide an algorithm that solves the approximation problem (see Definition 6).

4.3.1 Influence Maximization is NP-hard

Theorem 1. *The Influence Maximization Problem is NP-hard.*

Proof. We follow the method of Jennifer Wortman [?], and prove this problem is NP-hard by reducing it to the set cover problem. In the set cover problem we are given a set $U = \{u_1, u_2, \dots, u_n\}$ and a collection of subsets $S_1, S_2, \dots, S_m$ and a integer k . We wish to determine whether there exist k subsets such that their union is the whole set U .

Given an instance of the set cover problem, we construct an instance of the activation maximization problem (which is an HSCM) as follows. First, create an empty graph $G = (V, E)$. Then construct $V = (\cup i = 1^m S_i) \cup (\cup j = 1^n u_j)$. For each $i \in \{1, 2, \dots, m\}$ and each $j \in \{1, 2, \dots, n\}$ add $evivj$ to E if and only if $uj \in S_i$; we let $w(evivj) = 1$ for all i, j .

Suppose now that we have an algorithm, which can solve the activation maximization problem. Thus, we can assume that we have the subset of V , T , where $|T| = k$, such that the expected size of the set of vertices activated is maximized after the 1st time step. Since the weight of each edge in E is 1, an edge between $v_1, v_2 \in V$ lets us conclude that v_1 activates v_2 given that v_1 is active. This lets us determine the expected size of set activated by T , denoted $\sigma(T)$, where $\sigma : V \rightarrow \mathbb{R}$ is the influence function. We do this by performing breadth-first searches¹ starting from each element in T , while maintaining a record of the nodes encountered by the search, adding to it new vertices when they are discovered, and finally returning the size of this record.

Finally, we consider whether $\sigma(T) \geq n+k$. Suppose that indeed $\sigma(T, 1) \geq n+k$. Since the nodes corresponding to the subsets S_i have no incoming edges, we expect all k of them will be active since they were selected to be members of T . In order to get n more nodes to be active, because the nodes corresponding to the subsets S_i do not have edges between one-another, all n nodes corresponding to each $u_i \in U$ must be active. Thus there exist some k subsets $S_1, S_2, \dots, S_k$ that cover U . Conversely, if $\sigma(W, 1) < n+k$ then even though k subsets are active, even the activation maximizing solution W is not able to choose k subsets that cover U . □

¹We terminate each breadth-first search upon reach a node with no outgoing edges, and we are guaranteed that there are no cycles in the graph G we constructed.

4.3.2 General Lower Bound on Approximation

Since the exact maximization problem is NP-hard, our goal is to come up with an approximation algorithm, to which we can assign a lower bound. Kempe [KKT05] introduces the following theorem.

Theorem 2. *Let f be a non-negative, monotone, submodular function on sets.*

1. *The greedy algorithm, which always picks the element v with largest marginal gain $f(S \cup \{v\}) - f(S)$, is a $(1 - 1/e)$ -approximation algorithm for maximizing f on k -element set S .*
2. *A greedy algorithm which always picks an element v within $1 - \epsilon$ of the largest marginal gain results in a $1 - 1/e - \epsilon'$ approximation, for some ϵ' depending polynomially on ϵ .*

We apply this theorem to σ . We know σ is monotone increasing from [Fos08], who proved it with the notion of absorbing processes in Markov chains, and because we will prove it is submodular below, we know that the following greedy algorithm provides solution that is at least $1 - 1/e - \epsilon$ of the optimal solution, where ϵ is a polynomially related to the precision of our approximation to VAP .

Algorithm 2 Maximum Influencing Set of Size k for Cascade of t Steps

Require: int $k \geq 1$, int $t \geq 1$, Graph G

```

 $res \leftarrow \{\}$ 
for  $i = 1$  to  $k$  do
   $maxVal \leftarrow 0$ 
   $v^* \leftarrow null$ 
  for all  $v \in G.V - res$  do
    if  $\sigma(res \cup v) \geq maxVal$  then
       $maxVal \leftarrow \sigma approx(res \cup v, t)$ 
       $v^* \leftarrow v$ 
    end if
  end for
   $res \leftarrow res \cup v^*$ 
end for
return  $res$ 

```

Recall that Algorithm 1 is an $O(It|V||E|)$ algorithm for approximating σ . Thus the greedy algorithm runs in $O(kIt|V|^2|E|)$.

4.4 Submodularity of Influence

Theorem 3. *The influence function, $\sigma(A, t)$, is submodular for all instances of the HSCM.*

Proof. Consider the case when we wish to find the influence of a set A for an HSCM represented by the graph G for t time steps. Next, consider an edge from a node in A , v to an inactive node u that has weight $w(evu)$. Since we are determining the influence of A , we let all nodes it contains be initially active. Then the probability that v activates u after t time steps is $1 - (1 - w(evu))^t$. Notice that this is the same as if there were t edges from v to u , where each could only try to activate u once. Using this idea, $\sigma(A, t)$ can be expressed in terms of the Independent Cascade Model's influence function introduced by [KKT03].

Consider an arbitrary node v . If the shortest path from a node in A to v contains d edges, then the first time step tj for which $P(G, v, A, tj) > 0$ is $tj = d$. Consider a node u for which v lies on the shortest path to a node in A . It follows that $P(G, u, A, tu) > 0$ for $tu = tj + 1$. Now, consider a particular execution of the HSCM, if v is activated on the tj th time step, then it will have $t - tj$ chances to activate u . However, unless $v \in A$, it could happen that v gets activated at a later time step, and in that case it will have less chances to activate u .

We introduce the random variable X , which encodes a particular execution of the HSCM as an Independent Cascade Model. The possible executions can be enumerated by first considering all the cases of when nodes with $d = 1$ get activated, and for each of those all the cases when nodes with $d = 2$ get activated, and so on until $d = t - 1$. In each execution Xi , if a node v gets activated at tv , we add all of v 's edges to the graph represented by Xi $t - tv$ times; thus we simulate the execution of the HSCM where v gets $t - tv$ opportunities to active its descendants.

If we let $\sigma I(A)$ denote the influence function for the Independent Cascade Model, then we can express

$$\sigma(A, t) = \sum Xi P(Xi) \sigma I(A) \quad (6)$$

Since $\sigma I(A)$ is proven to be submodular in [KKT05] and since a linear combination of submodular functions is also submodular, it follows that $\sigma(A, t)$ is submodular. □

4.5 A Lower Bound on the Accuracy of the Greedy Approximation Algorithm

In this section we look at how our approximation algorithm for *VAP* affects the lower bound to the greedy approximation algorithm.

Definition 8. Bonferroni Inequality If for some mean μs and some interval $I s$, $P(\mu s \in I s) = 1 - \alpha s$, then for n means $P(\mu 1 \in I 1 \wedge \mu 2 \in I 2 \wedge \dots \wedge \mu n \in I n) = 1 - \sum i = 1^n \alpha i$. See [Law07].

Suppose we have a graph with 10,000 nodes, and we wish to achieve a confidence level of 90 percent ($\alpha = 0.1$) that the lower bound of the greedy algorithm is $1 - 1/e$. By the Bonferroni Inequality we need αs such that $0.9 = 1 - 10000\alpha s$, and we get $\alpha s = 1/100000$. Then let $\beta = 100$, or some other value, and perform $N(\alpha s, \beta)$ samples. Then continue taking samples such that the highest sample lies in an interval that does not overlap with any other interval anything less than completely (we assume round-off error is negligible). At this point we are 90 percent certain that each mean lies in the intervals we have derived from sampling, and thus we are 90 percent certain that at any stage of selecting vertex $1, \dots, k$ as a member of the optimal set, we select the vertex with actually the highest marginal influence. Thus overall we are 0.9^k percent certain that we have a $1 - 1/e$ approximation, and for example if $k = 30$, $\alpha = 0.9965$ and $\alpha s = 3.5/10000000$. With such low αs , in most graphs $N(\alpha, \beta)$ will be enormous, and not polynomial in the accuracy of the greedy approximation.

5 Experiments

5.1 Testing the *VAP* Approximation Algorithm

In this experiment we test the results we achieved in Section 4.2.

5.1.1 Experiment Definition and Hypothesis

Definition 9. Given a sample mean $\bar{X}i$, and a interval half length β , if we compute $N(\alpha, \beta)$ samples, how many percent of 500 tries will satisfy the condition $\bar{X}i - \beta \leq \mu i \leq \bar{X}i + \beta$?

The hope is that this percentage will be greater than $(1 - \alpha)100$ percent. Failure will suggest that our model is skewed from the normal distribution, and more than $N(\alpha, \beta)$ samples are necessary to achieve a specific confidence interval. In this case we will refer back to [Law07]. Furthermore, we can

only perform this experiment on small graphs (up to 11-12 nodes), because calculating μi means solving VAP exactly, which we do in exponential time.

5.1.2 Procedure and Results

Since we needed to know μi , we worked with a graph consisting of 11 nodes, and added edges from all nodes i to nodes j iff $i < j$, in total we had 55 edges. We let the weight of each edge be 0.23. We let the set of initially active nodes be $A = \{0\}$. We then asked, given this graph, what is the probability of node 10 being activated after 4 time steps. The exact VAP solver returned the value 0.979016(...).

We then turned to the bounding problem, and let $\alpha = 0.05$ and $\beta = 0.005$. This means we were looking at the number of runs necessary in order to produce results that were in the interval $I = (0.974, 0.984)$ 95 percent of the time. For each try (‘try’ as in Definition 9) we first took 1000 samples, and then kept taking samples until Equation 4 was satisfied. The number of samples that ended up being taken per try was in the range from 2500 to 4000.

The main result is that out of 500 tests, 479 were in the interval I . Thus, experimentally, $P(\bar{X}i \in I) = 0.958$, which is better than the 0.95 we hoped for. This suggests that the distribution of the sample means is skewed towards normal, and $N(\alpha, \beta)$ should be sufficient to achieve a $\pm\beta$ interval with $(1 - \alpha)100$ percent certainty.

5.2 Maximization Approximation Problem

In this section we compare the greedy algorithm against other commonly used heuristics.

The *sum of weights* heuristic computes the sum of the weights of all outgoing edges for all nodes and chooses the node with the highest sum.

The *highest degree* heuristic chooses the node with the highest number of outgoing edges.

The *central distance* heuristic first determines the average distance from each node to all other nodes and then ranks the nodes by increasing distance. The idea is that nodes with short distance to all other nodes are “central” to the network and thus are more influential. Since the graph is not fully connected, we just assign some very big number for the distance when there is no path between two nodes.

Random refers to an algorithm that picks nodes uniformly at random. In this case, the representative result is given by the average of 30 runs of

the algorithm.

It is trivial to make the greedy algorithm parallel. Recall that for each additional node we want to add to the target set, the algorithm has to go through all the nodes and pick the one giving the highest marginal gain if activated. This computation can be broken up in multiple parts in such a way to allow different computers to process different ranges of nodes and thus covering the whole network.

Our implementation consists of a server and clients where the server tells each client to perform the same task but for different interval of nodes. After handing out the tasks, the server waits for all machines to complete their computations and receives a single node (along with its gain) from each of them. It then compares these gains, picks the best node, and activates it for the next run. The process continues as many times as is the size of the target set we want to have.

Each evaluation of the gain from some node involves determining the value of the influence function (see Section 9.9.9) applied to the set containing this node. Since computing this is difficult by itself, we determine it using a simulation. The simulation is run certain number of times and the average is taken as the representative value when evaluating candidate nodes. The time it takes for the simulation to complete depends on the density of the network, so we decided to run it for 1000 times for the first experiment and 100 times for the rest. Now, we should stress that in the worst case the decreased accuracy of the simulation can result in unstable node selections (e.g. picking different nodes for each run if they have sufficiently close expected gains), but the expected number of active nodes (size of active set) is preserved. This makes sense because in all experiments we are interested in the size of the target set, not in which nodes it particularly contains.

5.2.1 Small Network, Random Weights

In this experiment we use a graph of 1000 nodes and 5000 edges randomly connecting them. Each edge is also assigned a random weight. The time step is set to 3.

Results are shown on Figure 1 and we can clearly see that the greedy algorithm beats all other heuristics by a wide margin. We can also observe that the sum of weights heuristic outperforms highest degree. This is because it is essentially an improved version of highest degree – taking into consideration not only the number of edges, but also the specific property (weight) of each one. Finally, we observe that the algorithm, which selects nodes randomly performs worse than all other heuristics for all experiments;

this is because the random algorithm does not consider any properties of the network.

Figure 1 also intuitively confirms the submodularity of the influence function. As we can see the performance of the greedy algorithm and all heuristics is increasing at decreasing rate. This is always the case, but is especially easy to see here – because of the relatively small network the “capacity” for increase is limited and gets smaller after each activated node. Observe that it takes around 40 nodes to activate more than 85% of the network in three steps.

Figure 1: 1000 nodes, 5000 links network with random weights

5.2.2 Large Network, Random Weights

Figure 2 shows an essentially the same experiment, but for a much bigger network – 10000 nodes and 50000 links. Time step is again 3. Here we can clearly see that the highest degree heuristic outperforms the centrality heuristic. In a bigger network, the number of outgoing edges for a node is a much more reliable indicator than the distance from the node to other nodes.

Note that the general order of the performance for all algorithms is preserved in this experiment.

We also see that the graphs are increasing but at a much lower rate. This is because the “capacity” of each newly activated node to activate others is not limited early on as it is in the smaller network. It takes many more nodes before the effect of adding a new node starts to vanish visibly.

Figure 2: 10000 nodes, 50000 links network with random weights

5.2.3 Large Network, Constant Weights (0.2)

We also performed an experiment on a random 10000 nodes, 50000 links network with constant weights set to (0.2). In this case (Figure 3) it turned out that the greedy algorithm performs slightly worse than the highest degree heuristic. We think that the advantage of the greedy algorithm is that it can actually “look ahead” a few nodes when selecting a node. But while this context proves helpful in the previous experiments, here it is almost useless in comparison to just looking at the degree: in this case you cannot learn much more by simulating the process that by merely looking at the outgoing degree of the nodes.

Figure 3: 10000 nodes, 50000 links network with constant weights of 0.2

Figure 4 shows how the number of active nodes is related to the time step. The network we used is the same as in the previous experiment (Figure 3) and the size of the target set is fixed to 25. We can see that the greedy algorithm and the highest degree heuristic have almost indistinguishable performance. We speculate that in the long run this will stay the same or greedy will very slightly outperform highest degree.

6 Future Work

Work related to the History Sensitive Cascade Model can continue along two branches, which we describe here. In the first section, we suggest important work that needs to be done in order to further complete the model. In the second section we consider applications to social data, and the problems that could be investigated.

6.1 Work on the Model

There are several interesting problems related to the HSCM model in-and-of itself.

Figure 4: 10000 nodes, 50000 links network with constant weights of 0.2, size of target set is 25

6.1.1 $VAP \stackrel{?}{\in} P$

The most interesting problem in this section is determining whether $VAP \stackrel{?}{\in} P$. We have attempted several reductions to known NP-hard problems and encountered problems,

- In reducing VAP to inference in Bayesian networks (which was proven NP-hard by Cooper in [Coo90]) we stumbled on the fact that Bayesian networks have more specification (We recommend Charniak's introduction to Bayesian Networks [Cha91]). For example, in a Bayesian a node can be activated with an arbitrary probability for any combination of its parents being active, whereas in our model these probabilities are implicit in the edges between nodes and their weights.
- In reducing VAP to 3SAT, we encountered the problem that we could not represent variables taking on true/false values, and as a result we could not avoid that if one clause was (x, x, x) and another was $(\neg x, \neg x, \neg x)$, both clauses could be activated by our model, which is clearly not the case in 3SAT.

- Finally we tried to reduce *VAP* to finding a path of length k in general graphs. Where the reduction broke down was when the only path of length k would be between two vertices, but there was a shorter path between the vertices as well.

We also tried to develop a polynomial time solution that would use a depth first search to find all paths from active vertices to a given vertex, and calculate that a path would activate the given vertex, and consider all paths. The problem with this approach is that there are possible exponentially many paths between two vertices. We suggest considering reductions to other NP-hard problems, and a possible dynamic programming solution.

6.1.2 A Better Lower Bound to the Greedy Approximation

The second problem that is related to the model, is coming up with better results than we were able to come up with in Section 4.5. The goal is to provide a lower bound, where the accuracy of the greedy influence maximization solver is polynomially related to the number of samples taken when finding the vertex with the greatest marginal gain. We recommend referring either to the original proof of the lower bound to the greedy algorithm in [NW78] or a more general version in [GS07], we also recommend [Law07].

6.1.3 Approaching *VAP* From Another Angle

We have chosen to work with *VAP* as defined, and the influence maximization problem. However, we can also look at this problem in terms of time minimization. The optimization problem then becomes to determine the set of k nodes such that the time t is minimized for which the expected number of nodes activated is some value c . The corresponding version of *VAP* would be to ask what is the time step t such that $P(G, v, A, t) \geq \theta$ for some $\theta \in (0, 1)$.

6.2 Applications to Social Data

Showing that the model is applicable to interesting problems in sociology and economics amplifies the significance of the above problems. We present several possible applications of the model.

6.2.1 Decision Making with the Recognition Heuristic

Goldstein and Gigerenzer from the Max Planck Institute for Human Development introduce the *recognition heuristic* in [GG02]. They use empirical

evidence to prove that the heuristic is ecologically rational, fast, frugal, and psychologically plausible model of human decision making in certain environments.

The recognition heuristic can be described as following:

If one of two objects is recognized and the other is not, then infer that the recognized object has the higher value with respect to the criterion.

The authors claim that the heuristic makes use of the structure of the environment. Goldstein and Gigerenzer explain how it works:

There are “mediators” in the environment that have the dual property of reflecting (but not revealing) the criterion and being accessible to the senses. For example, a person may have no direct information about the endowments of universities, because this information is often confidential. However, the endowment of a university may be reflected in how often it is mentioned in the newspaper. The more often a name occurs in the newspaper, the more likely it is that a person will hear of this name. Because the newspaper serves as a mediator, a person can make an inference about the inaccessible endowment criterion.

In their paper they talk about *ecological correlation* and *surrogate correlation* and maybe these can be somehow encoded in our model. Thus it can be used to analyze the environment someone lives in and how information propagates there.

6.2.2 Congestion Analysis

Our model might be useful for analyzing congestion in some networks. For example the routers and connections among them can be represented as vertices and edges. Then routers send data to another routers until the latter are flooded and brought down (*activated*). The progressiveness property ensures that brought down once, routers cannot be miraculously recovered.

6.2.3 Facebook Fashions

Using the data set about a college network on Facebook.com collected by KM Lewis, et al in [LKG⁺08], we propose a method for verifying Georg Simmel and Thörsten Vebeln’s theories about taste diffusion. The theories can be coarsely summarized by the statement that new trends and fashions

arise as higher social classes aim to differentiate themselves from lower social classes, while lower social classes try to emulate the higher ones. If we let high social class be approximated by popularity (because a popular person also may be emulated), this can be approximated by numbers of incoming friendships in KM Lewis et al's data. Then we could see how popular nodes in the network acquiring new fashions (eg by listing a film, artist, or novel among their favorites) leads to the spread of this 'fashion' to other nodes in the network.

The main problem we foresee with this is that the sample data provided by KM Lewis et al covers a network of less than 1700 college students, and it is clear that each of these students experiences significant influence from outside the network. Furthermore, we cannot be certain about the extent and accuracy of the tastes that students disclose, simply because only about 60 percent of KM Lewis et al's sample had any tastes disclosed at all, and there is no way of knowing if they had any tastes that they kept to themselves.

References

- [Cha91] Eugene Charniak. Bayesian networks without tears. *AI Magazine Volume 12 Number 4*, 1991.
- [Chu74] K L Chung. *A Course on Probability Theory*. 1974.
- [Coo90] Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 1990.
- [DR01] Pedro Domingos and Matt Richardson. Mining the network value of customers. *ACM*, 2001.
- [Fos08] Stephen Foster. Trinity reu. 2008.
- [GG02] DG Goldstein and G Gigerenzer. Models of ecological rationality: The recognition heuristic. *PSYCHOLOGICAL REVIEW-NEW YORK-*, 109(1):75–90, 2002.
- [Gla00] Malcolm Gladwell. *The Tipping Point*. Little Brown, 2000.
- [GS07] PR. Goundan and AS. Schulz. Revisiting the greedy approach to submodular set function maximization. 2007.

- [KKT03] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD*, 2003.
- [KKT05] D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, pages 1127–1138. Springer Verlag, 2005.
- [Law07] Averill M. Law. *Simulation & Modeling Analysis*. McGraw Hill, 2007.
- [LKG⁺08] Kevin Lewis, Jason Kaufman, Marco Gonzalez, Andreas Wimmer, and Nicholas Christakis. Tastes, ties, and time: A new social network dataset using facebook.com. *Social Networks*, In Press, Accepted Manuscript, 2008.
- [MR07] Elchanan Mossel and Sebastien Roch. On the submodularity of influence in social networks. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 128–134, New York, NY, USA, 2007. ACM.
- [NW78] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *MATHEMATICS OF OPERATIONS RESEARCH*, 3(3):177–188, 1978.
- [Sur04] James Surowiecki. *The Wisdom of Crowds*. Random House, 2004.
- [Wor08] Jennifer Wortman. Viral marketing and the diffusion of trends on social networks. Technical report, 2008.
- [Zha09] Yu Zhang. A deterministic model for history sensitive cascade in diffusion networks. *SMC*, 2009.