Heuristics for Advertising Revenue Optimization in Online Social Networks

Inzamam Rahaman and Patrick Hosein
Department of Computer Science
The University of the West Indies
St. Augustine, Trinidad
inzamam@lab.tt, patrick.hosein@sta.uwi.edu

***Abstract*—Recent increases in the adoption of Online Social Networks (OSNs) for advertising has resulted in the research and development of algorithms that can maximize the resulting revenue. OSN users are likely to be influenced by their friends; therefore, one can leverage friendship relationships to determine how advertisements should be distributed among users. If a user is given an indication that their friend clicked on an advertisement link (called an impression), then they are more likely to also click on the impression if it were to be provided to them. The problem of assigning impressions can be modeled as an optimization problem in which the goal is to maximize the expected number of clicks achieved given a fixed number of impressions. Hosein and Lawrence [1] formulated this as a Stochastic Dynamic Programming problem in which impressions are provided in stages and the outcomes of previous stages are used in making impression allocations for the present stage. However, the determination of the optimal solution is computationally intractable for large problems; hence we require heuristics that are efficient while providing near-optimal solutions. In this paper we provide and compare various heuristics for this problem.**

***Index Terms*—Online Social Networks, Optimization, Dynamic Programming**

I. Introduction

With the advent of low cost smart phones, the adoption rate of the Internet continues to increase globally. One particular application that is quite popular is Online Social Networking (OSN) such as Facebook [2]. This has lead to companies allocating an increasingly greater portion of their advertising budget to providing advertisements on these OSNs [3]. Many companies such as Google and Facebook derive the majority of their revenue by providing this service to companies. Such companies must determine the optimal placement, timing and target audience for these advertisements. In [1], a Stochastic Dynamic Programming model is provided for this problem which we follow in this paper.

The model can be described as follows. A company pays on a per impression basis for a given number of impressions for its advertisements to be placed on the OSN. These impressions are placed in stages. In each stage, a subset of the impressions are allocated and the outcomes of these allocations (whether or not the user clicked on the impression) are observed before allocations are made in the subsequent stage. The decision of each user given an impression (whether or not they clicked) is made available to the user's friends to whom impressions are subsequently allocated so that they can take that information into account. Typically, if one's friend clicked an impression then one is more likely to also click; consequently, the probability of clicking increases accordingly. Therefore, impressions are placed in each stage such that the number of clicks generated is maximized for a fixed number of impressions. Moreover, in addition to allocating impressions in each stage, one must also determine how many impressions should be allocated in each stage. The number of clicks reflects the number of users who may eventually purchase the products and the resultant revenue derived by the company.

A. Related Work and Contributions

This paper is based on the model proposed by Hosein et. al. [1]. In their model they considered a multi-stage deployment. Prior work in this area considered the problem of influence maximization [4], [5], [6], [7], [8] which can be considered as a special case of a single stage of the formulation in [1]. In [1] the model was presented and the optimal solution derived. They also included a heuristic but no performance comparisons were made. Our contributions include additional heuristics for the problem as well as performance comparisons. The proposed heuristics are fast enough to be used in a practical deployment while providing near-optimal performance.

One difficulty in these influence models as well as the model in [1] is the determination of the influence probabilities between parties. In other words, how much are we affected by the actions of our friends. In the paper by Lei et. al. [9] feedback is used to update influence information. Several heuristics have also been developed for the influence maximization model such as Prefix Excluding Maximum Influence Path [10] and Influence Ranking and Influence Estimation [11]. In our work we use a simple influence function that was used in [1] since we believe that the relative performance of the various proposed heuristics will not be affected by the specific function used. We plan to do further work in this area.

The major contributions of this paper are the proposed heuristics and the evaluation of these heuristics to demonstrate their performance and computational efficiency. In the next section we provide the mathematical model that is used. We then provide details of the proposed heuristics followed by numerical results.

IEEE/ACM ASONAM 2016, August 18-21
2016, San Francisco, CA, USA
978-1-5090-2846-7/16/$31.00 © 2016 IEEE

II. Mathematical Model

OSNs can be represented as graphs. A graph comprises a set of vertices, also called nodes, denoted by the set V and a set of edges denoted by the set E. The vertices represent users, and the edges represent a friendship relationship between a pair of users. We assume that friendships are reciprocal. Consequently, the graph representing the OSN is taken as undirected. Note that this is not the case for some OSNs such as Twitter. We denote the number of users in the OSN by $N = |V|$.

In Hosein and Lawrence's [1] model, friendships are taken as exerting influence (positive or negative) on a user's probability of clicking an impression. Suppose that u_1 and u_2 are friends. If u_1 clicks in a previous stage, then the probability of u_2 clicking in the present stage should increase. Likewise, if u_1 does not click when given an impression in a prior stage, then the probability of u_2 clicking in the present stage will, in general, decrease. Since all friendships are assumed to have the same degree of influence, the same influence function is used for all pairs. Hence, the change in a user's probability of clicking is related to the number of their friends who have clicked an impression and the number of their friends who did not click. We use the following function to update the probability of clicking given information of the user's friends actions in prior stages.

$$p \leftarrow min\ \{1,\ max\ \{0,\ p_{init} + \alpha\frac{f}{n} - \beta\frac{g}{n}\}\} \tag{1}$$

where n is the number friends, p_{init} is the initial probability, f is the number of friends that were provided impressions and clicked and g is the number of friends that did not click their allocated impression. The factors $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ determine the degree of influence. In most realistic situations, negative information (i.e. the fact that a friend did not click) will typically not be provided since it is not in the provider's best interest to do so. This can be modeled by setting $\beta = 0$, which we do for our numerical examples.

Impressions are allocated in batches at the start of a stage. There is a set interval between stages in which a user's response to an impression is recorded. Both the number of impressions to be distributed during an instance of the problem, M, and the number of stages, K, are assumed given. The optimization problem must be solved in each stage. However, the number of impressions used in each stage must also be determined. These are represented by the vector m indexed 1 through K where m_{K-k} is used to indicate the number of impressions to use in the present stage with $K-k$ stages to go. Note that the determining optimal $\vec{m}$ is also a difficult problem, so we also introduce an efficient heuristic for determining this vector. Moreover, all impressions need not be used; however, since the objective function is non-decreasing with respect to the number of impressions, the the optimal solution will contain the case in which all impressions are used. Therefore, we will assume that $\sum_{k=0}^{K-1} m_k = M$.

Let index k denote the number of stages to go such that in the last stage $k = 0$ while in the first stage $k = K-1$. For each value of k, the current state may be described by two vectors: $\vec{x}$, which is used to denote whether or not a user was previously given an impression; and $\vec{c}$, which is used to denote whether a user clicked on a prior impression. In addition, the vector $\vec{u}$, the decision vector, denotes whether or not an impression is to be allocated in the present stage and the vector $\vec{p}$ represents the probability of clicking. For each user i, these vectors are updated in stage k as follows:

$$x_k[i] = 1 \text{ if } i \text{ was previously given an impression else it is } 0$$

$$c_k[i] = 1 \text{ if } i \text{ clicked a given past impression else it is } 0$$

$$u_k[i] = 1 \text{ if } i \text{ is given an impression in this stage else it is } 0$$

$$p_k[i] = \begin{cases} \text{prob}(c_{k-1}[i] = 1 | u_k[i] = 1, \vec{x}_k, \vec{c}_k) & \text{if } x_k[i] = 0 \\ 0 & \text{otherwise} \end{cases}$$

For each stage, the expected number of clicks is obtained by summing over all possible outcomes: for each outcome, the probability of that outcome times the expected number of clicks obtained from that outcome. This is determined by the expected outcome from the allocations in the next stage. We let the set of all possible outcomes be $\mathcal{V} \subseteq \{0,1\}^{|V|}$. Suppose $\vec{v} \in \mathcal{V}$, then $v[i] = 0$ if $u_k[i] = 0$ (since a click cannot occur without an impression) and $v[i] = 0$ or 1 if $u_k[i] = 1$. Therefore, $|\mathcal{V}| = 2^{m_k}$ since m_k impressions are provided in this stage. The probability that $\vec{v}$ occurs is given by

$$Pr(\vec{v}) = \prod_{i=1}^{N} u_k[i]\{p_k[i]v[i] + (1-p_k[i])(1-v[i])\} + 1 - u_k[i] \tag{2}$$

For a given allocation, $\vec{u}$, and outcome $\vec{v}$ we update $\vec{x}_{k-1} = \vec{x}_k + \vec{u}_k$ since new impressions have been allocated and $\vec{c}_{k-1} = \vec{c}_k + \vec{v}$ to account for new clicks.

If we let $J^*_{k-1}(\vec{x}_{k-1}, \vec{c}_{k-1}, \vec{p}_{k-1})$ be the optimal expected number of clicks in the subsequent stage, then we may write the optimal expected value for some stage k as:

$$J^*_k = \max_{\vec{u} \in \{0,1\}^N} \sum_{\vec{v} \in \mathcal{V} | \vec{u}} Pr(\vec{v}) J^*_{k-1}(\vec{x}^k + \vec{u}, \vec{c}_k + \vec{v}, \vec{p}_{k-1}) \tag{3}$$

$$\text{subject to: } \sum_{i=1}^{N} u[i] = m_k \quad \text{and} \quad \vec{u} + \vec{x}_k \leq 1.$$

In the final stage we have:

$$J^*_0 = |\vec{c}_0| + \max_{\vec{u} \in \{0,1\}^N} \sum_{i=1}^{N} p_0[i]u[i] \tag{4}$$

$$\text{subject to: } \sum_{i=1}^{N} u[i] = m_0 \text{ and } \vec{u} + \vec{x}_0 \leq 1.$$

and in this case the solution is simply the sum of the m_0 largest probabilities. If we denote this optimal allocation by $\vec{u}^*$ then:

$$J^*_0 = |\vec{c}_0| + \sum_{i=1}^{N} p_0[i]u^*[i] \tag{5}$$

III. Heuristics

Determining the optimal solution of the problem formulated in Hosein and Lawrence [1] is computationally intensive. With k stages to go, there are $\binom{n}{m_k}$ possible ways of distributing the impressions for that stage where n is the number of users who have not yet been given an impression. For each such event, a user given an impression may or may not click and hence 2^{m_k} possible outcomes must be evaluated. Thus, $\binom{n}{m_k}2^{m_k}$ sub-problems must be solved when $k-1$ stages are left. Since OSNs can be very large (on the order of millions of users), heuristics must be used. One such heuristic was proposed (but not evaluated) in Hosein and Lawrence [1]. In this section, we consider that heuristic together with some of our own designs to determine what will work best in a practical setting.

A. The Hosein-Lawrence Heuristic

Consider k stages to go with n users who are possible candidates for impression assignments and m_k impressions to assign. We need to identify users $u_1, u_2, \ldots, u_{m_k}$ that would yield the optimal expected number of clicks. First, we identify the user u_1 whom, with no other impressions allocated in this stage, would be the best single candidate for an impression. After identifying u_1, we similarly identify u_2. However, along with our candidates for u_2, we also allocate an impression to the previously identified u_1. After finding the user u_2 that yields the highest expected number of clicks in concert with u_1, we continue to proceed in this manner until we have identified the user u_{m_k}. The impressions for this stage are then assigned to users u_1 through u_{m_k}. Since the expected number of clicks in the final stage does not rely on any subsequent stage's results, the last stage procedure is reduced to identifying the m_0 users that have not be assigned impressions who exhibit the highest probabilities of clicking.

This heuristic was analyzed in [1] for the two stage problem. They show that, if we denote the computational complexity of the optimal solution by C_{opt} (in terms of number of single stage problems that must be solved) and that of the heuristic by C_{h1} then the ratio of these complexities grow as follows:

$$\frac{C_{opt}}{C_{h1}} = \mathcal{O}\left(\frac{1}{n}\binom{n}{m}\right) \tag{6}$$

where n users can be given impressions and m impressions are to be allocated in the present stage.

B. The Maximum Influence (MI) Heuristic

Note that, at each stage, the probability of clicking is updated for each user. In addition, we know who has already been provided impressions in the past. In the present stage, for each user who has not yet been given an impression, let $p[i]$ denote the clicking probability of user i. Furthermore let $d[i]$ denote the number of i's friends who have not yet been given an impression. Suppose we wanted to find the user who, if given an impression, provides the largest influence in the subsequent stage. One metric that can be used to measure this is the product $p[i]d[i]$. In other words, as $p[i]$ increases then it is more likely that the user will click and influence the next stage and as $d[i]$ increases then the impact of this influence (number of affected users) also increases. Therefore this heuristic is as follows, in the present stage we compute $p[i]d[i]$ for each user and assign impressions to those with the largest products. Note that both $p[i]$ and $d[i]$ must be updated anyway and so the additional computation due to this heuristic is minimal. Furthermore, note that the computational complexity is independent of the number of stages. At each stage there is only one possible set of users to whom impressions will be allocated. The users to be allocated impressions is found as previously described and once these are known the number of subproblems equals the number of possible outcomes 2^{m_k}.

C. Local Search and Monte Carlo Simulation (LSMC)

Note that there is a finite number of possible allocations of impressions to users. Therefore we can potentially evaluate each of these and choose the one that provides the largest expected number of clicks. However, for large networks there are two issues. Firstly the number of possible impression allocations is large and secondly the evaluation of the expected number of clicks for a given allocation is also computationally intensive. We overcome these by (a) using a local search approach over the impression allocation space and (b) using Monte Carlo simulations to approximate the expected number of clicks for a given allocation.

Suppose we start with some allocation of impressions to users in each stage. We can compute the expected number of clicks for this allocation to determine how well it performs but this is computationally intensive. We instead use Monte Carlo simulations to get a reasonable estimate for the allocation. In the first stage, we randomly generate outcomes of the allocated impressions using the click probabilities in the first stage. We then compute the click probabilities in the second stage and again randomly generate the outcomes for the impressions allocated in the second stage based on the computed click probabilities. We repeat this process for all stages and determine the number of clicks obtained. This is repeated several times and the average taken to represent an estimate of the expected number of clicks for the given allocation.

We choose a new allocation to evaluate as follows. We compute click probabilities for each user for the present allocation while performing the above Monte Carlo Simulations and take their average. The new allocation is obtained by removing the impression for the user with the lowest click probability and providing it the user with no impression with the highest click probability. One issue with this approach is that the average click probability increases with each stage. To overcome this we can normalize the probabilities in each stage so that the average click probability over all users in each stage is the same. After this is done we then do the comparisons for the swap. Hence, using these two approaches we can reduce the run time taken to get a reasonable estimate of the optimal value. However we can get as close as we desire by (a) increasing the number of allocations we search and (b) by increasing the number of Monte Carlo evaluations.

TABLE I
DATASET PARAMETERS

Data	Users	Impressions	Stages	Avg No Friends	p_{K-1}
1	6	5	2	2.7	0.25
2	7	5	3	2.0	0.25
3	15	7	3	4.4	0.25
4	50	10	3	3.5	0.25
5	100	20	3	5.5	0.25
6	1000	20	2	100	0.20

IV. NUMERICAL RESULTS

A. Problem Datasets

To evaluate the performance and runtime of the heuristics against the optimal solution, each method was implemented in Python and run on a server with 2GB of RAM and a 1.8 GHz processor. We considered six datasets of varying sizes as described in table I. Across the datasets we varied the number of users, the number of impressions, the number of stages and the network properties. Datasets 3 and 6 were constructed using the Erdős-Rényi [12] model. In this model, we use two parameters to generate a random graph, the number of nodes, and the probability of an edge existing between two arbitrary nodes. For each pair of nodes we randomly determine whether or not an edge exists between them [13]. Probabilities of 0.26 and 0.11 were used to generate datasets 3 and 6 respectively. Datasets 4 and 5 were sampled from the Stanford Network Analysis Project's Facebook dataset [14] to generate a graph with 50 and 100 users respectively. For each dataset, the function used to update a user's probability of clicking is as described in 1 with $\alpha = 0.25$ and $\beta = 0$. We set the initial click probability, p_{K-1} to 0.25 for datasets 1-5 and 0.2 for dataset 6.

B. Performance and Runtime Comparisons

In this section we provide a comparison of the performance results for the six datasets. For datasets 1-3 the optimal solution is computed while LSMC evaluations are used for comparison in the remaining datasets. The results are provided in Table II. In Table II, we also provide the impression vector that produced the optimal solution. These were obtained by evaluating all possible impression vectors. Note that the Hosein-Lawrence heuristic require impractical runtimes for datasets 4-6 and so were not determined.

Another important criterion is the heuristic runtimes. In a real system the solution will have to be computed for much larger network. Hence, low runtimes are essential. In Table II we also provide the runtime values for the different datasets for each method. Note that the LSMC values are just for the time taken to reach the stopping criterion we used. As seen in Table II, the time taken by the optimal formulation grows quickly as the size of the OSN grows, highlighting the need for heuristics. Although the Hosein-Lawrence heuristic performs well, its run time increases quickly with problem size; thus it is not suitable for large problems. On the other hand the Maximum Influence Heuristic provides acceptable values with reasonable runtimes even for large problems. It should be noted that for dataset 5, LSMC outperformed the Maximum Influence Heuristic and so can potentially be used as a heuristic. However, it also suffers from poor runtimes on large problems.

TABLE II
PERFORMANCE AND RUNTIME COMPARISONS

Data	Method	Optimal $\vec{m}$	Value	Time (ms)
1	Optimal	[2, 3]	1.40	64
	Hosein and Lawrence	[2, 3]	1.40	32
	Maximum Influence	[3, 2]	1.38	8
2	Optimal	[2, 2, 1]	1.56	56841
	Hosein and Lawrence	[1, 2, 2]	1.54	4678
	Maximum Influence	[1, 2, 2]	1.52	48
3	Optimal	[2, 2, 3]	2.03	170967777
	Hosein and Lawrence	[2, 2, 3]	2.03	211516
	Maximum Influence	[2, 2, 3]	1.99	378
4	LSMC	[2, 2, 6]	3.22	7229250
	Maximum Influence	[2, 1, 7]	3.09	10444
5	LSMC	[2, 2, 16]	6.54	23426641
	Maximum Influence	[2, 3, 15]	6.39	40097214
6	LSMC	[10, 10]	4.32	302874000
	Maximum Influence	[8, 12]	4.04	49838538

V. OPTIMAL NUMBER OF IMPRESSIONS PER STAGE

In the mathematical formulation in Section II we assumed that the number of impressions per stage $\vec{m}$ was given and solved for the optimal impression allocations at each stage. So to obtain the optimal solution we need to evaluate all possible impression vector allocations as we did in the previous section for the exact computation of the optimal solution. As we saw previously, even when the impression vector is given, the determination of the optimal solution is computationally intensive. In this section we discuss heuristics for the impression vector determination. Such a heuristic will remove the need to evaluate all possible impression vectors.

We propose the following heuristic for determining the number of impressions to use in each stage. Consider a symmetric graph in which all nodes have the same degree d. Denote the click probability, for all users, in the first stage by p_K (i.e. a $K + 1$ stage problem). No matter how the impressions are placed, the expected number of clicks will be $p_K m_K$ since m_K impressions are allocated in the first stage. Each of the successful clicks will cause an increase in the click probability of all friends of the user who clicked. Note that a user may have more than one friend who clicked, but this will be an unlikely case. Consequently, we will assume that users have at most one friend who clicked. Therefore, the expected number of users for whom a single friend clicked is given by $p_K m_K d$. Now we need to have m_K large enough to increase the click probability of sufficient friends for the subsequent stage. However, if m_K is too large then there will be some users with increased click probabilities but these cannot be given impressions in the subsequent stage because there are not enough impressions. Therefore we choose m_K such that the expected number of users with increased probabilities equals the number of impressions available in the next stage.

$$p_K m_K d = m_{K-1} \quad (7)$$

Note that, in reality, all nodes do not have the same degree (i.e. all users do not have the same number of friends) but we can take d to be the average node degree. In the next stage we assume that only the users that have increased click probabilities are considered for impressions. Hence their click probabilities are now $p_K + \alpha/d$ since we assume one friend has clicked and we are assuming $\beta = 0$. However note that, an impression given to one of these users now only affects $d-1$ friends since one of their friends had clicked. So in the second stage we have

$$p_{K-1} m_{K-1}(d-1) = m_{K-2} \quad (8)$$

We can repeat this all the way to the last stage and then use the fact that the total number of impressions must equal M to determine the number of impressions in each stage.

For example let us consider the case of $K = 3$ We have

$$m_2 = \frac{m_1}{p_2 d} \quad \text{and} \quad m_1 = \frac{m_0}{p_1(d-1)} \quad (9)$$

Since the total number of impressions is M we have

$$M = m_0 + m_1 + m_2 = m_0 \left[1 + \frac{1}{p_1(d-1)} + \frac{1}{p_2 p_1 d(d-1)}\right]$$

Therefore

$$m_0 = \frac{M}{1 + \frac{1}{p_1(d-1)} + \frac{1}{p_2 p_1 d(d-1)}} \quad \text{where} \quad p_1 = p_2 + \frac{\alpha}{d} \quad (10)$$

since we assume that in each stage a single friend of a user clicks. Note that, when computing the probabilities, appropriate checks must be made to ensure that the probability values do not exceed unity. Given m_0 we can now use 9 to determine the impressions for the first two stages.

In Table III, we compare the performance of this impression vector heuristic with the optimal solution. For each of the datasets we provide the optimal impression vector and the optimal value in column 2 while in column 3 we provide the impression vector obtained by the heuristic together with the optimal function value that is obtained for this impression vector. We find that the heuristic impression vector is close to the optimal one and for the cases where it is not close the objective function values are comparable (which means that the objective function value is not sensitive to the impression vector for these cases).

TABLE III
PERFORMANCE OF IMPRESSION VECTOR HEURISTIC

Dataset	Optimal Vector (value)	Heuristic Vector (value)
1	[2, 3] (1.40)	[2, 3] (1.40)
2	[2, 2, 1] (1.56)	[3, 1, 1] (1.51)
3	[2, 2, 3] (2.03)	[2, 2, 3] (2.03)
4	[2, 2, 6] (3.22)	[3, 3, 4] (3.14)
5	[2, 2, 16] (6.54)	[4, 6, 10] (6.41)
6	[10, 10] (4.32)	[1, 19] (4.22)

VI. Conclusion and Future Work

Advertising in Online Social Networks is a quite lucrative business; hence algorithms that can provide near optimal results in near real-time are important. We considered the revenue optimization model that was introduced in [1] and developed heuristics that provide good results while not being too heavily computationally intensive. Due to the time taken to evaluate large problems we focused on smaller illustrative examples that show the proposed Maximum Influence heuristic performed quite well and was fast. We also introduced a heuristic for determining how many impressions should be used in each stage of the problem. We believe that a combination of these two heuristics will perform well in real OSNs.

References

[1] P. Hosein and T. Lawrence, "Stochastic dynamic programming model for revenue optimization in social networks," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on*, Oct 2015, pp. 378–383.

[2] (2014) Facebook doubleclick for publishers (dfp) optimization website. [Online]. Available: https://www.facebook.com/business/a/online-sales/ad-optimization-measurement

[3] (2013) Iabinternet advertising revenue report. [Online]. Available: http://www.iab.net/media/file/ IABInternetAdvertisingRevenueReport-tHY2013FINALdoc.pdf

[4] D. Kempe, J. Kleinberg, and E. 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*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 137–146.

[5] E. Bakshy, D. Eckles, R. Yan, and I. Rosenn, "Social influence in social advertising: Evidence from field experiments," in *Proceedings of the 13th ACM Conference on Electronic Commerce*. ACM, 2012, pp. 146–161.

[6] H. Bao and E. Y. Chang, "Adheat: An influence-based diffusion model for propagating hints to match ads," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/1772690.1772699

[7] S. Bhagat, A. Goyal, and L. V. Lakshmanan, "Maximizing product adoption in social networks," in *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, ser. WSDM '12. New York, NY, USA: ACM, 2012, pp. 603–612. [Online]. Available: http://doi.acm.org/10.1145/2124295.2124368

[8] J. Hartline, V. Mirrokni, and M. Sundararajan, "Optimal marketing strategies over social networks," in *Proceedings of the 17th International Conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 189–198. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367524

[9] S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart, "Online influence maximization," in *Proc. KDD*, Sydney, Australia, Aug. 2015.

[10] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. New York, NY, USA: ACM, 2010, pp. 1029–1038. [Online]. Available: http://doi.acm.org/10.1145/1835804.1835934

[11] K. Jung, W. Heo, and W. Chen, "Irie: Scalable and robust influence maximization in social networks," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 918–923.

[12] A. Erdős, P.; Rényi, "On random graphs. i," *Publicationes Mathematicae*, pp. 290–297, 1959.

[13] E. N. Gilbert, "Random graphs," *Ann. Math. Statist.*, vol. 30, no. 4, pp. 1141–1144, 12 1959. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177706098

[14] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.