On the Multi-Stage Influence Maximization Problem

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*—The influence maximization problem turns up in many online social networks (OSN) in which each participant can potentially influence the decisions made by others in the network. Relationships can be friendships, family relationships or even professional relationships. Such influences can be used to achieve some objective and the influence maximization problem attempts to make decisions so as to maximize the effect of these influences. Past work focused on a static problem whereby one tries to identify the participants who are the most influential. Recently, a multi-stage version of the problem was proposed in which outcomes of influence attempts are observed before additional participants are chosen. For example, in online advertising, one can provide an impression to a particular subject and if that subject clicks on the impression, then their friends are informed in the hope that this information will increase the chances that they also click on the impression and eventually purchase the product. This problem is computationally intensive; in this paper we investigate various optimization methods for finding its solution that yield close to optimal results while taking less computation time. These include greedy and particle swarm optimization algorithms.**

***Index Terms*—Online Social Networks, Dynamic Programming, Influence Maximization, Particle Swarm Optimization**

I. Introduction

With the rapid growth of the Internet and the reduction in the cost of the devices used to access it (i.e., smart phones, tablets, etc.) more people are going online to keep in touch with friends, family and colleagues. One particular application that is quite popular is Online Social Networking (OSN) such as Facebook [1]. This trend has lead to companies allocating an increasingly greater portion of their advertising budget to providing advertisements on these OSNs [2]. In addition, many social experiments are now possible because one can more easily collect and analyze data. In such networks, each participant is related in some way to a subset of the other participants. They may be friends, relatives, office colleagues or even professional acquaintances. Therefore one participant may be able to influence others on matters such as product purchases, services and politics. If one were interested in disseminating information (e.g., about a product) then one can more efficiently do so by first convincing an influential person of the product's value and then have the person influence others. This Influence Maximization problem has been well studied. In [3] a multi-stage version of this problem was introduced and this further increases the computational complexity of the problem.

978-1-5090-5105-2/16/$31.00

If we consider an advertising campaign, a company pays 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 is 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 member given an impression is made available to the member's friends so that they can take that information into account if subsequently provided with an impression. Typically, if one's friend clicked an impression then one is more likely to also click; hence the probability of clicking increases accordingly. The objective is therefore to place impressions in each stage such that the total number of clicks is maximized. Note that, in addition to making impression allocations in each stage, one must also determine how many impressions should be used in each stage. The number of clicks reflects the number of users who may eventually purchase the product and hence the revenue derived by the company. In this paper we follow the formulation of the problem as presented in [3] although, as we mentioned previously, this is just one example of influence maximization. In the next section we provide the mathematical model and then we describe different optimization methods that can be used to solve it.

A. Related Work and Contributions

This paper is based on the model proposed by Hosein and Lawrence [3]. 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 the formulation in [3].

One difficulty in these influence models as well as the model in [3] 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].

The major contributions of this paper are the proposed optimization methods and a comparison of these methods. We use a modified version of the cascade model [12] for the influence probability and describe how we determined the relevant parameters for the model. In the next section we provide the mathematical model that is used, followed

by details of the optimization methods and finally numerical results including some parameter sensitivity analyses.

II. Mathematical Model

An OSN can be represented as a graph consisting of a set of nodes, denoted by the set V, and a set of edges denoted by the set E, where $E \subseteq V \times V$. The nodes represent OSN members and the edges represent a friendship relationship between a pair of members. Friendships in OSNs are reciprocal and so the graph representing an OSN is undirected. We shall denote the number of users in the OSN by N and hence $N = |V|$. The set of friends of user i will be denoted by F_i. We first describe the influence probability model that is used. This determines how one member's decision affects their friends' decisions. We then describe the model used to determine the impression allocation strategy that maximizes the expected number of clicks.

Impressions are allocated in batches at the start of a stage. There is a set interval between stages in which a member's response to an impression is recorded. The total number of impressions is denoted by M and the number of stages by K. We must determine to whom impressions should be allocated and in which stage the allocation to that member should be made. Note that the optimization problem must be re-solved in each stage since the outcomes of the previous stage should be taken into account. This problem was formulated as a Stochastic Dynamic Programming Problem in [3]. The optimal solution can be obtained but that requires an exhaustive search. In this paper we provide a different formulation of the problem.

A. Influence Probability

In the model in [3], friendships are taken as exerting influence (positive or negative) on a user's probability of clicking an impression. Suppose that i and j are friends. If i clicks in a previous stage, then the probability that j clicks in the present stage should increase. Likewise, if i does not click when given an impression in a prior stage then the probability of j clicking in the present stage should decrease. All friendships are assumed to have the same degree of influence and so the same influence probability function is used for all pairs. Note that a member's probability of clicking is related to the number of their friends who have clicked (or not clicked) in the past.

The papers [13], [14] and [12] focus on two main influence models, the Linear Threshold (LT) Model and the Independent Cascade (IC) Model. In the LT model, whether a participant clicks or not depends on the subset of friends who have already clicked. The joint influence probability of the set of friends who have clicked is determined and if this exceeds some threshold then we assume the member also clicks. In the IC model each friend of a member independently influences the member with some given probability. The joint probability is then used as the probability of clicking for the member.

We follow the Independent Cascade model but with two modifications from the model provided in [13]. Firstly, in the multi-stage case the influence probability in the first stage (i.e. no friends have yet clicked) must be non-zero. Therefore we assume that the click probabilities of all users in the first stage is p_0. Secondly, we use the fact that a person who has a lot of friends in the social network will typically be influenced less by any one of them than a user with a few (close) friends. Therefore we assume that the probability that an a person is influenced by someone who has clicked decreases with the number of friends of the person. Finally let z denote all friends who have previously clicked and let F denote the total number of friends. We use the following click probability for the member:

$$p = 1 - (1 - p_0)\left(1 - \frac{\alpha}{F}\right)^z \tag{1}$$

The factor $0 \leq \alpha \leq F$ determines the degree of influence in the particular OSN. The choice of p_0 depends on the initial interest of the product to the member. We use $p_0 = 0.05$ in our numerical results which indicates that a member is interested in one out of twenty products without any influences. The parameter α depends on how many people it takes before we are almost convinced that we should click on an impression. We did an informal survey and determined that a suitable value for this was $\alpha = 10$.

B. Impression Allocation Space

We assume that there are K stages which are indexed by k. Let x_{ik} denote the allocation vector whereby $x_{ik} = 1$ if member i is allocated an impression in stage k and 0 otherwise. Note that there is a finite number of possible allocations. Each user can be allocated an impression at most once and the total number of allocations over all stages must equal M. We can compute the number of possible allocations as follows. The number of possible allocation of impressions to members is simply N choose M. However the allocation to a member can be made in any of the K stages and there are K^M ways to do this. Of course we need to remove the cases in which no allocations are made in a stage but since we typically have $K << M$ then the number of those cases is insignificant. Therefore the number of allocations that must be evaluated is given by $\binom{N}{M}K^M$ which can be extremely large.

Instead of evaluating all possible allocations we will instead use a greedy algorithm that starts from a single impression allocation and sequentially adds impressions to the user/stage pair that provides the largest increase in the objective function value.

C. Objective Function Evaluation

For each of these allocations we must evaluate the expected number of clicks. For a given allocation, the expected number of clicks is the sum over all members of the expected number of clicks of each member. The expected number of clicks of a member is the probability of clicking, when assigned an impression, times 1 and hence we simply need to compute the click probability of each user and take the sum. However, these probabilities vary based on what allocations were previously made. Let p_{ik} denote the probability of clicking of user i in

stage k given some feasible allocation $\mathbf{X}$. If no impression was given to the user in stage k then this probability is simply 0. Note that in stage 1 we have $pi1 = p_0$ for all users i since $z = 0$ in 1. We now compute p_{ik}, if an impression is given in stage k, in terms of probabilities in earlier stages. Consider user i and let C_i denote the subset of their friends who have been given impressions in prior stages. For simplicity, for each friend $j \in C_i$ define

$$q_j = \sum_{k=1}^{k-1} x_{jk} p_{jk}.$$

This simply means that once an impression is provided to a user then the probability that they clicked is the clicking probability of the user in the stage in which the impression was given. Let $\mathcal{V}$ denote the set of all vectors of size $|C_i|$ such that each member is either 0 or 1.

Therefore $\mathcal{V}$ represents all possible outcomes of allocations made to member i's friends in previous stages. We can then compute the clicking probability of user i as

$$p_{ik} = \sum_{\vec{v} \in \mathcal{V}} P(\vec{v}) \prod_{j=1}^{|C_i|} \{v[j] q_j + (1 - v[j])(1 - q_j)\} \quad (2)$$

where we define

$$P(\vec{v}) = 1 - (1 - p_0) \left(1 - \frac{\alpha}{F_i}\right)^{\sum_i v_i} \quad (3)$$

and where F_i is the total number of friends of user i. Once we compute all click probabilities then the resulting objective function value is simply given by

$$J(\mathbf{X}) = \sum_{k=1}^{K} \sum_{i=1}^{N} x_{ik} p_{ik} \quad (4)$$

D. Optimization Problem

The optimization problem can now be stated as finding the allocation vector that provides the largest expected number of clicks and hence we have:

$$\max_{\mathbf{X}} J(\mathbf{X}) \quad (5)$$

$$\text{s.t.} \quad \sum_{k=1}^{K} x_{ik} \leq 1 \quad \forall i \qquad \text{and} \qquad \sum_{k=1}^{K} \sum_{i=1}^{N} x_{ik} = M$$

$$\text{with} \quad x_{ik} \in \{0, 1\} \quad \forall i, k$$

The first constraint is due to the fact that at most one impression can be allocated to a user. The second constraint is due to the fact that the total number of allocated impressions is M. Finally this is an integer programming problem since an impression can either be allocated or not allocated.

One can therefore find the optimal solution by evaluating $J(\mathbf{X})$ for all feasible values of $\mathbf{X}$ and choosing the allocation with the highest objective function value. Unfortunately (a) the number of possible allocations is very large and (b) the evaluation of the expectation for each allocation is computationally intensive. In the next section we describe ways to address these two issues.

III. Optimization Methods

A. Greedy Algorithm (GA)

We first provide an approximation to the computation of the expected number of clicks for a given allocation and then use this to do a greedy allocation of impressions. Consider some initial feasible allocation $\mathbf{X}$. In the first stage the click probabilities, p_0, are known and so we use these as our approximations in the first stage. Let us assume that the approximate click probabilities are known for stages 1 to k and determine the click probability approximations for stage $k+1$. Consider some user i and let C_i denote the set of friends who have so far been given impressions. Let $\hat{p}_j$ denote the approximate click probability of friend $j \in C_i$ in the stage in which the impression was given. Therefore the expected total number of clicks performed by these friends is given by

$$T_i = \sum_{j \in C_i} \hat{p}_j \quad (6)$$

We use this number to compute the approximate click probability of user i in the present stage (if given an impression) by

$$\hat{p}_{ik+1} = 1 - (1 - p_0) \left(1 - \frac{\alpha}{F_i}\right)^{T_i}. \quad (7)$$

Note that this computation can be done quite rapidly. We evaluated this approximation and found that it generally overestimates the true value (as one would expect since the influence probability function is concave). However, in terms of comparing two allocations it performs quite well. In other words if one allocation is better than another then this relation also holds, in general, for the approximation.

For a given allocation $\mathbf{X}$, let $\hat{J}(\mathbf{X})$ denote the above approximation to the expected number of clicks for that allocation. For any user j with no impression in this allocation let $\mathbf{X}_{+j}$ be used to denote the corresponding allocation in which an impression is also given to j. Therefore the expected value increase due to this allocation is given by

$$\Delta_j = \hat{J}(\mathbf{X}_{+j}) - \hat{J}(\mathbf{X}). \quad (8)$$

Note that Δ_j consists of two components, one component is the click probability of j and the second component is the effect on j's friends in later stages if j were to click. Furthermore, the former component increases with stages (i.e., as members click then this increases the clicking probabilities in later stages) while the latter component decreases with number of stages (i.e., as the number of stages left decreases, the future influence a click has will also decrease).

We use the following greedy approach. We assign the first impression by choosing the user who, if given an impression in the first stage, has the most influence. To measure influence in the first stage, we use Betweenness Centrality [15]. Assume we have already made k impression allocations. Given these allocations, in each stage we compute for each member (without an impression) their click probability (if given an impression). In each stage we normalize (over all members) the click probabilities so that the average is the same as

that in the first stage. This is done to compensate for the second component mentioned above. Having done this, we find the member and stage with the largest click probability and allocate an impression to that member in that stage. This is repeated until all M impressions are assigned.

B. Particle Swarm Optimization (PSO)

Initially developed by Kennedy and Eberhart [16], Particle Swarm Optimization (PSO) is a stochastic optimization method inspired by the swarm behaviors exhibited by birds and fish. In PSO, a fixed population of points in the search space are encoded as particles whose movement through the search space is influenced by both the flocking behavior of its peers and its own history of behavior. Initially, the particle represents a random point in the search space. From this initial point, the movement of a particle through the search space is influenced by a velocity that can be used to update the particle's position to a new position in the search space. To derive its velocity in a particular iteration, a particle must be cognizant of three facts: its velocity in previous iterations, its best position in the past, and the optimal solution among its peers. If U_j^i represents the velocity of the j^{th} particle in the i^{th} iteration, d_j is the j^{th} particle, prev_best_j is the previous best position of the j^{th} particle, global_best is the best particle thus far, $c_1, c_2, \omega \in [0, 1]$ are constants, and $r_1, r_2 \in [0, 1]$ are randomly generated, U_j^i is computed as follows

$$U' = c_1 r_1 \left(\text{prev\_best}_j - d_j\right) \tag{9}$$

$$U'' = c_2 r_2 \left(\text{global\_best} - d_j\right) \tag{10}$$

$$U_j^i = \omega U_j^{i-1} + U' + U'' \tag{11}$$

The process of particles navigating the search space continues until some stopping criterion is satisfied.

In its original form, PSO is designed to operate in a continuous space with particles and velocities for an n-dimensional search space encoded as vectors in $\mathbf{R}^n$. Consequently, PSO needs to be adapted to operate on a discrete search space. To this end, work such as Chen et al. [17], Akhand et al. [18], Pang et al. [19][20], Liu and Huang [21], and Zhang et al. [22] have adapted PSO to solve various discrete problems by proposing extensions to PSO and redefining components of PSO to suit the objective to optimize.

For our problem, our goal was to find the allocation $\mathbf{X}$ that maximizes J. Consequently, valid allocations constituted particles for our implementation of PSO. Given that the velocity is supposed to intelligently inform the generation of a new position from the current position, we determined the subtraction, addition, and scalar multiplication as canonically defined on matrices was ill-suited for generating new allocations. To this end, we redefined subtraction, addition, and multiplication to treat the matrices like fuzzy sets within the context of velocity computation similar to what is detailed in Chen et al. [10]. The definitions are as follows:

$$(\mathbf{X} - \mathbf{Y})_{ij} = \begin{cases} 0 & \text{if } \mathbf{X}_{ij} = 0 \\ 1 & \text{if } \mathbf{X}_{ij} = 1 \text{ and } \mathbf{Y}_{ij} = 0 \\ 2 & \text{otherwise} \end{cases} \tag{12}$$

$$(\mathbf{X} + \mathbf{Y})_{ij} = \begin{cases} \mathbf{X}_{ij} + \mathbf{Y}_{ij} & \text{if } \mathbf{X}_{ij} + \mathbf{Y}_{ij} \leq 1 \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

$$c\mathbf{X}_{ij} = \begin{cases} c\mathbf{X}_{ij} & \text{if } c\mathbf{X}_{ij} \leq 1 \\ 1 & \text{otherwise} \end{cases} \tag{14}$$

The above definitions are used to generate a $N \times K$ matrix that can then be used to inform the modification of a particle's position. Like in Chen et al. [17], we use the cut_θ operator to generate a crisp velocity. For every particle, a new value of θ is generated every iteration from a matrix in the space $\{0, 1\}^{N \times K}$ as described by:

$$\text{cut}_\theta\left(\mathbf{X}_{ij}\right) = \begin{cases} 1 & \text{if } \mathbf{X}_{ij} \geq \theta \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

This matrix is then used in conjunction with a particle's position to construct, by means of random sampling on the unified sets for each stage, a new position such that the invariant $\sum_{k=1}^{K} \sum_{i=1}^{N} X_{ik} = M$ and $\forall i \sum_{k=1}^{K} x_{ik} \leq 1$ where X is the particle holds. To further reduce computation time, the approximation $\hat{J}$ is used to compare particles and get an approximation of their objective value. However, once the optimal allocation is identified, the actual objective function value is computed using J.

IV. Numerical Results

We evaluated the accuracy and performance of the aforementioned optimization methods by running their Python 2.7 implementations on a server with 2GB of RAM and a 1.8 GHz processor. Numpy [23] and NetworkX [24] were used to provide numerical computing support, the betweeness centrality implementation, and a base for our own graph data structures. To evaluate performance, we used Python's built-in Timeit module. For the PSO, ω, c_1, c_2 and the default number of particles were set to 0.4, 0.5, 0.75, and 100 respectively.

A. Dataset Description

The Erdős-Rényi [25] model was used to generate the graphs used in our experiments. The Erdős-Rényi [25] model accepts two parameters, the number of nodes in the graph, N and the probability of an edge existing between two arbitrary nodes. Optimal values for datasets 1 and 2 were obtained by using the model detailed in Hosein and Lawrence [3]. However, due to the computational complexity of said optimal formulation, only PSO and GA were applied to datasets 3 and 4. For all datasets, the values for α and p_0 were set at 10 and 0.05 respectively.

TABLE I
DATASET PARAMETERS

Dataset	Users	Impressions	No. Stages	Avg. No. Friends
1	15	5	3	10.86
2	25	5	3	10.88
3	500	7	3	99.07
4	1000	7	3	387.90

TABLE II
PERFORMANCE AND RUNTIME COMPARISONS

Dataset	Method	Optimal $\vec{m}$	Value	Time (ms)
1	Optimal	[1, 2, 2]	2.47	755935
	Greedy Algorithm	[1, 2, 2]	2.41	81
	Particle Swarm	[1, 2, 2]	2.41	26861
2	Optimal	[2, 1, 2]	2.30	20919057
	Greedy Algorithm	[2, 1, 2]	2.20	110
	Particle Swarm	[2, 1, 2]	2.20	39575
3	Greedy Algorithm	[2, 3, 2]	2.16	505474
	Particle Swarm	[2, 2, 3]	2.23	581779
4	Greedy Algorithm	[3, 2, 2]	1.85	260739
	Particle Swarm	[2, 3, 2]	1.85	1049364

TABLE III
THE EFFECT OF K ON THE EXPECTED NUMBER OF CLICKS

K	Method	Optimal $\vec{m}$	Value	Time (ms)
3	Greedy Algorithm	[2, 3, 2]	2.16	505474
	Particle Swarm	[2, 2, 3]	2.23	581779
4	Greedy Algorithm	[1, 1, 3, 2]	2.22	2188357
	Particle Swarm	[2, 2, 0, 3]	2.23	2823510
5	Greedy Algorithm	[1, 2, 1, 3, 0]	2.26	9129112
	Particle Swarm	[1, 0, 1, 1, 4]	2.24	10181358

TABLE IV
THE EFFECT OF M ON THE EXPECTED NUMBER OF CLICKS

M	Method	Optimal $\vec{m}$	Value	Time (ms)
7	Greedy Algorithm	[2, 3, 2]	2.16	505474
	Particle Swarm	[2, 2, 3]	2.23	581779
8	Greedy Algorithm	[1, 4, 3]	2.51	610680
	Particle Swarm	[4, 1, 3]	2.49	803556
9	Greedy Algorithm	[2, 6, 1]	2.80	761927
	Particle Swarm	[2, 3, 4]	2.784	1052105
10	Greedy Algorithm	[3, 3, 4]	3.27	895593
	Particle Swarm	[5, 2, 3]	3.155	1323398

B. Results

As can be seen in Table II, the PSO and Greedy Algorithms reasonably approximated the Optimal expected number of clicks and correctly ascertained the optimal impression vector (i.e., the number of impressions in each stage). Furthermore, the PSO and GA both took substantially less time than computation of the optimal solution.

C. Sensitivity Analysis

Due to the computational complexity involved in determining or approximating the expected number of clicks resulting from an allocation of impressions to users, it is important to understand the degree to which additional stages or impressions can impact the expected number of clicks. To this end, we varied the number of stages and the number of impressions designated for dataset 3 and recorded the optimal values and time taken by both the PSO and GA. In addition, to analyze the impact of α on the expected number of clicks, we varied α and computed the expected number of clicks for 7 impressions allocated across 3 stages for dataset 4.

1) Dependence on Number of Stages: To compare the values obtained by varying the number of stages, we kept the number of impressions fixed at 7. Recall that, as the number of stages increases, so to does the number of possible allocations. For example with 3 stages and 7 impressions, there are 36 possible impression allocations. In contrast, with 5 stages and 7 impressions, there are 96 possible impression allocations. Consequently, as more combinations need to be considered, the computation time of both the GA and PSO would increase as we consider problems with more stages. This can be seen in Table III, where increasing the number of stages increases the computation time. Moreover, as the number of stages is increased, the additional value of each additional stage decreases. In light of the computation time incurred by increasing the number of stages, from these results, we can infer 3 stages would be adequate for most problems.

2) Dependence on Number of Impressions: To compare the values obtained by varying the number of impressions, we kept the number of stages fixed at 3. As can be seen in Table IV, adding more impressions can lead to a fair gain in the expected number of clicks. It should be noted, however, that increasing the number of impressions also increases the computation time as more impressions would lead to more potential combinations and a larger search space. This is highlighted by the fact that the PSO's computation time increases more steeply with respect to the number of impressions than the GA due to PSO being a meta-heuristic for searching the solution space of allocations. Furthermore, the GA tended to find better allocations than the PSO for a larger number of impressions. This can be remedied by having a larger initial population of particles. However, by increasing the number of particles, this would further increase the computation time of the PSO.

3) Dependence on the Influence Probability Parameters: To ascertain the impact of α on the expected number of clicks, we varied the values α from 8 to 12 and computed the expected number of clicks. The results of this experiment can be seen in Table V. We find that the value of α had a small effect on the expected number of clicks. As α increases, the expected number of clicks also slightly increases. Recall that $\left(1-\frac{\alpha}{N}\right)^z$ decreases as $\frac{\alpha}{N}$ increases. Consequently, $(1-p_0)\left(1-\frac{\alpha}{N}\right)^z$ would decrease as $\frac{\alpha}{N}$ increases. Thus $1-(1-p_0)\left(1-\frac{\alpha}{N}\right)^z$ increases as $\frac{\alpha}{N}$ increases, thereby leading to a higher probability of clicking and a higher expected number of clicks.

TABLE V
THE EFFECT OF α ON THE EXPECTED NUMBER OF CLICKS

α	Method	Optimal $\vec{m}$	Value	Time (ms)
8	Greedy Algorithm	[2, 2, 3]	1.82	337762
	Particle Swarm	[2, 2, 3]	1.82	1065990
9	Greedy Algorithm	[2, 3, 2]	1.83	322050
	Particle Swarm	[2, 1, 2]	1.83	1068036
10	Greedy Algorithm	[3, 2, 2]	1.85	260739
	Particle Swarm	[2, 3, 2]	1.85	1049364
11	Greedy Algorithm	[2, 2, 3]	1.84	261751
	Particle Swarm	[2, 3, 2]	1.84	1057048
12	Greedy Algorithm	[3, 2, 2]	1.85	258480
	Particle Swarm	[1, 3, 3]	1.85	1054867

V. CONCLUSION AND FUTURE WORK

The problem of Influence Maximization is important since it can be used to model a wide range of problems in online social networks. In particular, the problem of impression placement so as to maximize the advertising revenue achieved. In this paper we addressed a multi-stage version of the Influence Maximization problem. We provided a new formulation of the problem and used two techniques for its solution, the traditional greedy algorithm and a Particle Swarm Optimization approach. These approaches were implemented and compared in terms of accuracy and computation run time. Run time is important since such problems need to typically be solved in near real-time. In future work we plan to derive an upper bound for the optimal value so that we can determine how well these methods perform in very large online social networks such as Facebook. We also plan to explore how techniques such as MapReduce [26] can be used to solve very large problems.

REFERENCES

[1] (2014) Facebook doubleclick for publishers (dfp) optimization website. [Online]. Available: https://www.facebook.com/business/a/online-sales/ad-optimization-measurement

[2] (2013) Iabinternet advertising revenue report. [Online]. Available: http://www.iab.net/media/file/ IABInternetAdvertisingRevenueReportHY2013FINALdoc.pdf

[3] 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.

[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] H. Narasimhan, D. C. Parkes, and Y. Singer, "Learnability of influence in networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 3168–3176.

[13] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 241–250.

[14] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," *Theory OF Computing*, vol. 11, no. 4, pp. 105–147, 2015.

[15] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, 1977.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[17] W. N. Chen, J. Zhang, H. S. H. Chung, W. L. Zhong, W. G. Wu, and Y. h. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 278–300, April 2010.

[18] M. A. H. Akhand, S. Akter, S. S. Rahman, and M. M. H. Rahman, "Particle swarm optimization with partial search to solve traveling salesman problem," in *Computer and Communication Engineering (ICCCE), 2012 International Conference on*, July 2012, pp. 118–121.

[19] W. Pang, K. ping Wang, C. guang Zhou, and L. jiang Dong, "Fuzzy discrete particle swarm optimization for solving traveling salesman problem," in *Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on*, Sept 2004, pp. 796–800.

[20] W. Pang, K.-P. Wang, C.-G. Zhou, L.-J. Dong, M. Liu, H.-Y. Zhang, and J.-Y. Wang, "Modified particle swarm optimization based on space transformation for solving traveling salesman problem," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 4, Aug 2004, pp. 2342–2346 vol.4.

[21] Z. Liu and L. Huang, "A mixed discrete particle swarm optimization for tsp," in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, vol. 2, Aug 2010, pp. V2–208–V2–211.

[22] C. Zhang, J. Sun, Y. Wang, and Q. Yang, "An improved discrete particle swarm optimization algorithm for tsp," in *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, ser. WI-IATW '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 35–38. [Online]. Available: http://dl.acm.org/citation.cfm?id=1339264.1339653

[23] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011. [Online]. Available: http://scitation.aip.org/content/aip/journal/cise/13/2/10.1109/MCSE.2011.37

[24] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

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

[26] J. Lin and M. Schatz, "Design patterns for efficient graph algorithms in mapreduce," in *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, ser. MLG '10. New York, NY, USA: ACM, 2010, pp. 78–85. [Online]. Available: http://doi.acm.org/10.1145/1830252.1830263