Using MapReduce for Impression Allocation in Online Social Networks [*]

Inzamam Rahaman
The University of the West Indies
St. Augustine, Trinidad
inzamam@lab.tt

Patrick Hosein
The University of the West Indies
St. Augustine, Trinidad
patrick.hosein@sta.uwi.edu

ABSTRACT
Online Social Networks (OSNs) offer an efficient and cost effective platform for the dissemination of advertisements to potential consumers. User actions and relationships can be analyzed to make informed decisions about where, how, and to whom advertisement impressions should be allocated to maximize the efficacy of an advertising campaign in an OSN. Based on the observation that users influence their friends, research has been done to use this information for determining impression allocations. Recently, a multistage formulation that involves the allocation of advertisements in stages, has been proposed as a stochastic dynamic programming problem. We have developed heuristics based on this formulation and believe that the MapReduce Programming model can be used to further increase performance and reduce compute time. This involves using cluster analysis together with a distributed version of our proposed greedy algorithm. In this paper we provide the framework for this distributed algorithm.

General Terms
Online Social Network, Optimization

Keywords
Online Social Network, Optimization, MapReduce, Dynamic Programming, Clustering, Distributed Computing

1. PROBLEM
An increase in the adoption of the Internet has led to an increase in the use of Online Social Networks (OSNs) such as Facebook [9] as a means for users to connect with friends, family, and colleagues. In response to this increase, many companies have allocated a large fraction of their advertising budget to advertising on social networks [13]. Consequently, work on using influence maximization [14, 2, 3, 5, 11, 1] has been undertaken to identify the users to whom advertisement impressions should be allocated.

[*]This is a position paper.

Hosein and Lawrence [12] developed a stochastic dynamic programming formulation of the problem of allocating impressions to users that aimed to maximize the expected number of clicks. This formulation differed most other solutions to the problem of impression allocated that approached the problem from the standpoint of Influence Maximization. The work by Abassi et al. [1] is the closest to that presented in Hosein and Lawrence, with the latter's model being a generalization of the former's.

In Hosein and Lawrence's formulation, advertisements, termed impressions, are allocated to a subset of N users in batches in stages separated by intervals of time. For a particular instance of the problem, the total number of impressions to allocate, M, the number of stages, K, and the number of impressions per stage, $\vec{m} \in \mathbb{Z}^K$ such that $\sum_{i=1}^{K} m\left[i\right] = M$, are fixed. In each stage, a user either clicks or does not click. Based on their actions, we then allocate impressions to users in the next stage. If a user being allocated an impression in the current stage had friends who clicked their impression in previous stages, said user is informed of their friends' decisions. This is modeled by a an influence function which is dependent on the number of a person's friends who have so far clicked. Since a user's decision to click an impression would be influenced by their friends' actions, their probability of clicking the impression themselves increases. To compute the expected number of clicks, we consider all possible outcomes at each stage and sum across all of them. To formalize the model briefly. Let k represent the number of stages to go such that $\vec{x}_k$, $\vec{c}_k$ and $\vec{u}_k \in \{0,1\}^N$ that represents whether or not users have been allocated impressions, users have clicked given an impression in the past, and users have been given an impression in the current stage respectively. Moreover, let $p_k \in [0,1]^N$ be the probability of clicking for users, and $J^*_{k-1}(\vec{x}_{k-1}, \vec{c}_{k-1}, \vec{p}_{k-1})$ be the optimal expected number of clicks in the subsequent stage. Then we may formulate the optimal expected value for some stage k as:

Table 1: Dataset Parameters

Data	Users	Impressions	Stages
1	6	5	2
2	7	5	3
3	15	7	3

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

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

Thus, in the final stage we have:

$$J_0^* = |\vec{c}_0| + \max_{\vec{u}\in\{0,1\}^N} \sum_{i=1}^{N} p_0[i]u[i] \quad (2)$$

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

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:

$$J_0^* = |\vec{c}_0| + \sum_{i=1}^{N} p_0[i]u^*[i] \quad (3)$$

At the k^{th} stage of Hosein and Lawrence's [12] formulation, 2^{m_k} sub-problems must be solved. Moreover, each sub-problem involves considering $\binom{N-\sum_{i=1}^{k-1}}{m_k}$ combinations of users at the k^{th} stage. Consequently, the optimal formulation is unsuitable for use in large OSNs. To remedy this, we developed and evaluated several heuristics to reduce the number of combinations that need to be considered to arrive at allocations that approach the optimal allocation in the expected number of clicks. We describe some of our heuristics below and outline some of our results on the datasets described in 1. Note that dataset 3 is sampled from the Stanford Network Analysis Facebook Dataset [15].

2. CLUSTERING HEURISTIC

Recall that in the model presented by Hosein and Lawrence, a user's probability of clicking is only perturbed from its default state by the actions taken by their friends to whom impressions were allocated. Many social networks can be divided into dense subgraphs where users are sparsely connected to users outside of said subgraph. These subgraphs are termed communities. [18, 17, 8, 20]. Due to the manner in which a user's probability of clicking is updated in Hosein and Lawrence's model, most changes to a user's probability would arise from the actions of users within the same community.

As most effects to a user's probability originates from inside their communities, we can divide the graph into communities and then assemble a subgraph using a subset of these communities to which we can then apply the optimal formulation. To partition the graph, $G = (V, E)$, into communities we used Louvain Modularity [8]. From Louvain Modularity, we derive communities $C_1, C_2, \ldots, C_p$. For each of these communities, we compute a subgraph $U_p = (C_p, \{(u, v) \mid \{(u, v) \in E, u, v \in C_p\})$. We then sort the subgraphs by descending order of average degree to get the ordered sequence $\langle U_1, U_2, \ldots, U_p\rangle$. To select the subsequence of subgraphs to generate our final subgraph, we find the smallest r such that

Table 2: Performance and Runtime Comparisons

Data	Method	Value	Time (ms)
1	Optimal	1.40	64
	Clustering Heuristic	1.40	66
2	Optimal	1.56	56841
	Clustering Heuristic	1.45	429
3	Optimal	2.03	170967777
	Clustering Heuristic	2.02	52401

$$\sum_{i=1}^{r} |U_i| \geqslant M \quad (4)$$

where $|U_i|$ refers to the number of verticies in the graph U_i. This integer r is then used to construct a subgraph of U' as follows:

$$U' = \bigcup_{i=1}^{r} U_i \quad (5)$$

After assembling U', we then applied the optimal formulation to U'. As seen in Table 2, the Clustering Heuristic performed well compared to the Optimal formulation, incurring a greater runtime for dataset 1 due to the lack of clusters in such a small dataset.

Since no two communities generated by Louvain Modularity would share users, the subgraph generated by the aforementioned procedure comprises of isolated islands of users. This, in conjunction with the results seen in Table 2, indicate that the most profitable users typically reside in these isolated islands of users that constitute the generated subgraph.

Consequently, by dividing a large OSN graph into mutually exclusive communities and rationing the number of impressions between them, we can split large instances of the problem into smaller instances. These smaller instances can then be solved in parallel with their results combined into the solution for the entire OSN graph. We believe that processing a large graph can be done by distributing communities among nodes in a computing clustering and using Parallelization Contracts [4], MapReduce [7], or message passing [16] to solve the expected number of clicks and allocations for each community.

3. PROPOSED METHOD FOR DISTRIBUTED COMPUTATION

Using community detection methods such as Louvain Modularity [8], Clauset et al. [6], Simulated Annealing [10], Pons and Latapy [19], and Wakira and Tsurumi [21], we can divide the graph into communities that can generate subgraphs that are smaller than the initial graph. Moreover, by using the average degree, we can select the subgraphs that are likely to be the most profitable. However, while there exists a clear way for the OSN graph to be divided into smaller units, i.e communities, for parallel processing, we need to determine a procedure for rationing the impressions between communities for the problem to be solved in a distributed manner.

To this end, we propose the following. Let $X \in [0,1]^{N\times K}$ such that $\sum_{i=1}^{N}\sum_{j=1}^{K} X_{ij} = M$ represents a valid allocation of impressions to users; if $X_{ij} = 1$, then user i is allocated an impression in stage j. Recall that a user's probability of clicking in a stage is affected by the number of their friends who, when given an impression in a previous stage, have already clicked. For every user who in the k^{th} stage had a friend who was given an impression in the $(k-1)^{\text{th}}$ stage, the expected number of their friends who clicked is simply the sum of the probabilities of said friends clicking in the $(k-1)^{\text{th}}$ stage. Moreover, the number of clicks generated in the k^{th} stage is the sum of the probability of clicking of the users allocated impressions in the k^{th} stage. Consequently, by applying this method, we can approximate the expected number of clicks generated by allocation X. For a given allocation X, let $\hat{J}(X)$ denote this approximation for X. We now use this approximation to the expected number of clicks evaluation to perform a greedy algorithm on each cluster as described below.

For any user j with no impression in allocation X, let 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

$$\Delta_j = \hat{J}(X_{+j}) - \hat{J}(X). \qquad (6)$$

We use a greedy approach to sequentially pick the user and stage that provides the largest Δ_j and allocate an impression to that user in that stage. This process is repeated until all M impressions are allocated. The initial user in this sequence is chosen as the one with the largest influence (obtained using the Betweenness Centrality metric). Note that the increment Δ_j decreases as the number of impressions are allocated and hence is concave. We use this fact in the distributed version of the problem since the number of impressions that should be allocated to each cluster should be such that they all have the same incremental value Δ otherwise an impression can be moved from a cluster with low increment (or gradient) to one with a higher gradient.

The above Greedy algorithm has been shown to perform well without clustering. In order to use it in a MapReduce Model we need to obtain a distributed implementation. This is done as follows. Let $\Delta_j(c)$ denote the incremental objective function value increase for providing an impression to user j who resides in cluster c. We simultaneously evaluate the optimal solution for each cluster in parallel. For each cluster we continue adding impressions until we obtain $\Delta_j(c) < \kappa$ for some threshold κ. Note that as κ is decreased then more impressions are included in each cluster and hence the total number of impressions allocated increases. Hence we need to find the appropriate value of κ to ensure that no more than M impressions (the budgeted amount) are used. This can be done by a simple message passing algorithm between cluster computations.

4. DISCUSSION

In this position paper we outline some of our findings thus far on using Hosein and Lawrence's [12] model for impression allocation in OSNs. These indicate the potential for community detection in dividing OSN graphs into subgraphs that can be solved independently during the map phase of MapReduce. Furthermore, we outlined a distributed greedy algorithm approach that can be used together with MapReduce. In this way the problem is first separated into clusters, each cluster is efficiently solved using a greedy algorithm and then the cluster solutions can be recombined to obtain the optimal solution. Future work will include a complete implementation of this framework using MapReduce.

5. ACKNOWLEDGMENTS

We acknowledge the Trinidad and Tobago Network Information Centre (TTNIC) for providing funding for this research.

6. REFERENCES

[1] Z. Abbassi, A. Bhaskara, and V. Misra. Optimizing display advertising in online social networks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1–11, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[2] 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*, pages 146–161. ACM, 2012.

[3] 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*, WWW '10, pages 71–80, New York, NY, USA, 2010. ACM.

[4] D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/pacts: A programming model and execution framework for web-scale analytical processing. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 119–130, New York, NY, USA, 2010. ACM.

[5] 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*, WSDM '12, pages 603–612, New York, NY, USA, 2012. ACM.

[6] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6), 2004.

[7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *6th Symposium on Operating System Design and Implementation*

(OSDI 2004), San Francisco, California, USA, December 6-8, 2004, pages 137–150, 2004.

[8] V. D. B. et. al. *Fast unfolding of communities in large networks*. Journal of Statistical Mechanics: Theory and Experiment, 2008.

[9] Facebook doubleclick for publishers (dfp) optimization website., 2014.

[10] R. GuimerÃă, M. Sales-Pardo, and L. A. N. Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical review E*, aug 2004.

[11] J. Hartline, V. Mirrokni, and M. Sundararajan. Optimal marketing strategies over social networks. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 189–198, New York, NY, USA, 2008. ACM.

[12] 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*, pages 378–383, Oct 2015.

[13] Iabinternet advertising revenue report., 2013.

[14] 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*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

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

[16] E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel Computing*, 22:789–828, 1996.

[17] M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B: Condensed Matter and Complex Systems*, 38(2):321–330, mar 2004.

[18] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[19] P. Pons and M. Latapy. *Computer and Information Sciences - ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings*, chapter Computing Communities in Large Networks Using Random Walks, pages 284–293. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[20] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, Jul 2006.

[21] K. Wakita and T. Tsurumi. Finding community structure in mega-scale social networks: [extended abstract]. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 1275–1276, New York, NY, USA, 2007. ACM.