MULTI-GRAPH SAMPLING OF ONLINE COMMUNITIES VIA MEAN HITTING TIME

Jacob Chakareski

Signal Processing Laboratory - LTS4, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

ABSTRACT

We derive a framework for sampling online communities based on the mean hitting time of its members, considering that there are multiple graphs associated with the same vertex set V representing the social network. First, we formulate random walk models on the multi-graph ensemble and define the essential properties of the mean hitting times associated with the corresponding Markov chains on the vertex set V. Then, we design a branch and bound optimization technique for computing the subset of vertices A that exhibits the shortest mean hitting time across the multi-graph, given a constraint on the size of A. We also design a greedy optimization method that computes an approximation to the optimal subset, at lower complexity, and that lends itself to a decentralized implementation, for further complexity reduction. We examine the performance of the sampling framework through a series of simulation experiments involving synthetic and actual samples of online community graphs. We demonstrate substantial improvements in terms of sampling (network) cost reduction and information dissemination speed relative to the state-of-the-art methods of node degree and eigenvector centrality.

1. INTRODUCTION

Frequently, one is faced with the task of selecting a representative subset of peers from an online community. Typically, these are the nodes (vertices) of the social graph that are the most influential within their community, frequently referred to as *information hubs*, and that therefore have a strong impact on the flow of information within the social network. Applications that can benefit from having such a subset of social peers are numerous and include recommendation systems [1], viral marketing [2], information search and retrieval [3], and medical and social studies [4], to name a few examples. In the present paper, we derive a formal framework for sampling online social networks for such nodes in the presence of multiple layers of graph information associated with the vertex set representing the online community.

For instance, beyond the actual topology of the social network, examples of such additional graph information include the data network connectivity graph between the social peers and the communication graph between them formed based on prospective privacy and security communication constraints that may exist. In fact, there is a range of topological layers of information that can be affiliated with the social network members based on the plethora of contextual information that is typically collected and generated today within and around online communities [5]. Taking advantage of a multi-graph ensemble can enhance the performance of information systems dealing with online communities, relative to the case when only the (single) social graph is considered, as our numerical experiments confirm. In particular, substantial improvements in terms of information dissemination speed and data transport efficiency are observed relative to the conventionally employed methods of selecting influential nodes based on various centrality measures associated with the social graph exclusively.

Our framework formalizes the concept of mean hitting time [6] over the multiple views of an online community, when selecting the most influential peers. In particular, due to the interaction of the multiple graphs of information, selecting the most influential members of the online community by solely considering social graph notions of node importance, may be quite under-performing when the application of such nodes is considered over the whole graph ensemble. For instance, it may lead to poor performance of viral marketing, characterized with slow and inefficient propagation of information. To overcome these shortcomings, our framework selects the most influential peers in an online community by jointly considering their mean hitting time impact across all available associated graphs.

We pose the problem of interest as constrained optimization of selecting a subset of nodes, whose size does not exceed a certain value, and whose mean hitting time is either minimal over a given graph or constrained not to exceed a predefined threshold. We derive a branch and bound technique for solving the optimization precisely. Its high complexity, however, precludes its application to real-life scenarios where the online communities can be exceedingly large in size. Hence, we also design a greedy and distributed technique that computes an approximation to the actual solution, at lower complexity. As our experiments show, the performance loss of the latter method is not dramatic in the case studies we considered, which makes it a suitable prospective candidate for actual deployment.

There have been a number of measures that have been proposed in the past to assess the centrality of a node's location within its community, e.g., its degree, betweenness, dominant eigenvector value, and closeness [7]. Among them, the eigenvector centrality has been the most widely, e.g., in the social sciences and information systems [8, 9]. By introducing the concepts of mean hitting time and multi-graph to the problem of node selection, our paper substantially advances the state-of-the-art, given the observed performance benefits.

2. RANDOM WALK MODEL

We assume that there are n views of the vertex set V characterizing an online community. Specifically, let $G_k = (V, A^{(k)}, W^{(k)})$ denote the k^{th} layer of graph information associated with the vertices in V, for $k = 1, \ldots, n$, where $A^{(k)}$ denotes the connectivity matrix of the graph and $W^{(k)} = [w_{ij}^{(k)}]$ denotes the corresponding matrix of edge weights. We index the social graph with $k = 1$, while the rest of the graph layers ($k > 1$) are constructed based on the additional contextual information that is available about the online community members.

With each graph G_k we associate a random walk that is constructed as follows. Let $\mathcal{N}_i^{(k)}$ denotes the set of neighbours of vertex i in the graph G_k. That is, these are the vertices $j \in V, j \neq i$ for which the entries $a_{ij}^{(k)} = 1$ in the corresponding matrix $A^{(k)}$. Next, let $w_i^{(k)}$ denote the overall weight of node i in G_k that is computed as $w_i^{(k)} = \sum_{j \in \mathcal{N}_i^{(k)}} w_{ij}^{(k)}$, where $w_{ij}^{(k)}$ is the i, j-th entry in the corre-

This work was supported by the Swiss National Science Foundation under Ambizione grant PZ00P2-126416.

sponding weight matrix $W^{(k)}$. Then, the transition probability $p_{ij}^{(k)}$ of the random walk on G_k can be computed as $p_{ij}^{(k)} = w_{ij}^{(k)}/w_i^{(k)}$, for $j \in \mathcal{N}_i^{(k)}$. Finally, $\pi_i^{(k)}$ denotes the stationary probability of the random walk on the graph G_k over the vertices $i \in V$ that is computed as $\pi_i^{(k)} = \frac{w_i^{(k)}}{\sum_{i \in S} w_i^{(k)}}$.

3. MEAN HITTING TIME

Let $(X_n)_{n \geq 0}$ denote a Markov chain characterized with transition probabilities p_{ij}, for $i, j \in S$, where S represents an indexing set of the states of the chain[1]. Let $A \subset S$ be a subset of the states of the chain. Then, with H^A we denote the first instance of time when the chain $(X_n)_{n \geq 0}$ reaches a state in A. Formally, H^A can be defined as $H^A = \inf\{n \geq 0 | X_n \in A\}$ and we refer to it henceforth as the hitting time of the chain for the subset A.

From the definition of the hitting time, it is obvious that H^A represents a random variable. Therefore, we define its expected value, conditioned on the fact that the chain started at state i, to be

$$\tau_i^A := \mathbb{E}(H^A | X_0 = i)\,. \tag{1}$$

To conserve space, we omit the proofs of the following propositions.

Proposition 1. *The mean hitting time is the smallest non negative solution of the linear system [10]*

$$\tau_i^A = \begin{cases} 0 & : \quad if \quad i \in A, \\ 1 + \sum_{j \in \mathcal{N}_i} p_{ij} \tau_j^A & : \quad otherwise, \end{cases} \tag{2}$$

where $\mathcal{N}_i$ denotes the set of neighbour states of i.

Proposition 2. *Let $\tau^A = (\tau_i^A)$ represent the vector of mean hitting times for each vertex $i \in S$. Then, τ^A satisfies the following matrix equation*

$$(I - P)\tau^A = \mathbf{1}_{|S|}\,, \tag{3}$$

where $P = [p_{ij}], i, j \in S$ is the matrix of transition probabilities of the chain $(X_n)_{n \geq 0}$, I is the identity matrix of size $|S| \times |S|$, and $\mathbf{1}_{|S|}$ is a (column) vector of ones of length $|S|$. Computing $\tau_i^A, i \notin A$, can be achieved via the following matrix inversion

$$\tau_e^A = (I - P_e)^{-1} \mathbf{1}_{|S|-|A|}\,, \tag{4}$$

where τ_e^A corresponds to the reduced vector of mean hitting times, with the entries τ_i^A in τ^A removed for indices $i \in A$. Similarly, P_e is a square matrix (of size $|S \setminus A| \times |S \setminus A|$)[2] that is obtained from P by eliminating the rows and columns that correspond to these indices. In this case, I represents the identity matrix of the same size as P_e.

4. HITTING TIME OPTIMIZATION

4.1. Problem Formulation

Let $A \in S$ represent a subset of nodes of the multi-graph. Let $\tau_{i,j}^A = \mathbb{E}_{G_j}(H^A | X_0 = i) = \mathbb{E}(H^A | X_0 = i, G_j)$ denote the mean hitting time for the set A on the graph G_j given that the corresponding Markov chain on G_j starts at state $i \in S$. Then, let $T_j^A = \{\tau_{1,j}^A, \ldots, \tau_{|S|,j}^A\}$ denote the ensemble of these hitting times for the vertex set S on the graph G_j, where $j = 1, \ldots, n$. Finally, we can compute the expected mean hitting time for the set A on the graph G_j as the expected value of T_j^A over the state set S. That is, we write $\mathbb{E}(T_j^A) = \sum_{i \in S} \tau_{i,j}^A \pi_i^{(j)}$, where $\pi^{(j)} = [\pi_i^{(j)}]$ represents the stationary distribution of the Markov chain on the graph G_j over the states $i \in S$.

We are interested in selecting a subset of nodes $A \subset S$ of a given size $|A|$ such that the average hitting time for the set A over the social graph G_1 is minimized. Simultaneously, A should be chosen such that the average hitting times for A over the rest of the graph ensemble, i.e., the graphs $G_2, \ldots, G_n$, should not exceed certain predefined values. Our goal is to balance the hitting time performance of our sampling approach across the multi-graph ensemble.

Formally, the problem under investigation can be written as

$$\begin{aligned} & \min_{A \in S} \mathbb{E}(T_1^A) \\ s.t. \quad & |A| \leq \kappa_1\,, \\ & \mathbb{E}(T_j^A) \leq \kappa_j \ \text{, for } j = 2, \ldots, n\,, \end{aligned} \tag{5}$$

where the quantities κ_j, for $j = 1, \ldots, n$, correspond to the constraints on the set size $|A|$ and the mean hitting times for the graphs $G_2, \ldots, G_n$, respectively.

Using the method of Lagrange multipliers, we can reformulate (5) as an unconstrained optimization

$$\min_{A \in S} L_{\lambda_1,\ldots,\lambda_n}(T_1^A, \ldots, T_n^A) = \mathbb{E}(T_1^A) + \lambda_1 |A| + \sum_{j=2}^{n} \lambda_j\, \mathbb{E}(T_j^A)\,. \tag{6}$$

where $\lambda_j > 0, j = 1, \ldots, n$, represent the corresponding Lagrange multipliers. Next, we design an optimization method for solving (6).

4.2. Branch and Bound Solution

Let $\mathbf{p} = (a_1, \ldots, a_{|\mathbf{p}|})$ represent a prefix of a given sequence of actions $\mathbf{a} = (a_1, \ldots, a_{|S|})$, for $|\mathbf{p}| \leq |S|$. Then, $\mathbb{E}(T_j^{A(\mathbf{a})}), \forall j$, will be smallest in value when $a_i = 1$, for $i = |\mathbf{p}| + 1, \ldots, |S|$. Therefore, $\epsilon_{\min,j}(\mathbf{p}) = \mathbb{E}(T_j^{A((\mathbf{p},1,\ldots,1))})$, for $j = 1, \ldots, n$, represent lower bounding values for the corresponding terms of the Lagrangian $L_{\lambda_1,\ldots,\lambda_n}(T_1^A, \ldots, T_n^A)$ in (6), for the given action prefix $\mathbf{p}$. Similarly, $\rho_{\min}(\mathbf{p}) = |A((\mathbf{p}, 0, \ldots, 0))|$ represents a lower bound for the second term in (6), again given $\mathbf{p}$. That is because the set $A(\mathbf{a})$ is smallest in size when the rest of the actions in $\mathbf{a}$, beyond the prefix $\mathbf{p}$, signify not to include in $A(\mathbf{a})$ the corresponding vertices $i \in S$, for $i = |\mathbf{p}| + 1, \ldots, |S|$. Hence, combining the bounding terms above provides us with an overall lower bound for the Lagrangian of any action vector $\mathbf{a}$ that includes $\mathbf{p}$ as its prefix. Formally, the bound can be written as

$$L_{\lambda,\min}(\mathbf{p}) = \epsilon_{\min,1}(\mathbf{p}) + \lambda_1\, \rho_{\min}(\mathbf{p}) + \sum_{j=2}^{n} \lambda_j\, \epsilon_{\min,j}(\mathbf{p})\,. \tag{7}$$

The algorithm starts with an arbitrary vector of actions $\mathbf{a} \in \{0, 1\}^{|S|}$. Alternatively, $\mathbf{a}$ can be initialized with an approximation of the solution of (6) that can be obtained ahead of time. The policy prefix $\mathbf{p}$ is empty originally. The algorithm then extends the prefix $\mathbf{p}$ to length $\text{len}(\mathbf{p}) + 1$, in a recursive fashion, until one of the following two conditions is met. Either the lower bound $L_{\lambda,\min}(\mathbf{p})$ exceeds $L_\lambda(\mathbf{a}) = L_{\lambda_1,\ldots,\lambda_n}(T_1^{A(\mathbf{a})}, \ldots, T_n^{A(\mathbf{a})})$ from (6), where $\mathbf{a}$ is the currently best vector of actions, or the length of $\mathbf{p}$ reaches $|S|$. In the latter case, $\mathbf{a}$ is updated with the action prefix of length $|S|$.

A recursion of the algorithm executes as follows. All possible extensions $\mathbf{p}_k$ of length one of the prefix $\mathbf{p}$ are determined. Given that we can append either zero or one to $\mathbf{p}$ (extensions of length one), there are only two possible extensions $\mathbf{p}_k$. We then calculate

[1] We omit reference to the graph index k in the exposition here. We refer to the vertex set V of the multi-graph as the set of states of the chain S.

[2] Note that "\" denotes the operator set difference.

$L_{\lambda,\min}(\mathbf{p}_k)$, for $k = 0, 1$, where $\mathbf{p}_0$ signifies the case when $\mathbf{p}$ is appended with zero and $\mathbf{p}_1$ denotes the respective extension with one. If $L_{\lambda,\min}(\mathbf{p}_k) > L_\lambda(\mathbf{a})$, no policy of prefix $\mathbf{p}_k$ can outperform the currently best policy $\mathbf{a}$. Otherwise, if $\text{len}(\mathbf{p}_k) = |S|$, the algorithm sets $\mathbf{a} = \mathbf{p}_k$ (a better action vector was found). On the other hand, if $\text{len}(\mathbf{p}_k) < |S|$, the recursion above is carried out again on $\mathbf{p}_k$. If the last condition is met by both $\mathbf{p}_k$ ($k = 0$ and $k = 1$), the recursion is carried out on each $\mathbf{p}_k$, in increasing order of their respective $L_{\lambda,\min}(\mathbf{p}_k)$ values. When the algorithm completes, $\mathbf{a}$ will comprise the optimal set of actions $\mathbf{a}^* = (a_1^*, \ldots, a_{|S|}^*)$. A formal description of the algorithm is provided below.

Algorithm 1 Branch and bound optimization

Initialize $\mathbf{a}, \mathbf{p} = ()$
BranchAndBound($\mathbf{p}$)

Recursive function: BranchAndBound($\mathbf{p}$)
Extend: $\mathbf{p}_k = (\mathbf{p}, k)$, for $k = 0, 1$
Compute: $L_{\lambda,\min}(\mathbf{p}_k)$, for $k = 0, 1$
if $L_{\lambda,\min}(\mathbf{p}_0) \leq L_{\lambda,\min}(\mathbf{p}_1)$ **then**
    ExaminePrefix($\mathbf{p}_0$); ExaminePrefix($\mathbf{p}_1$)
else
    ExaminePrefix($\mathbf{p}_1$); ExaminePrefix($\mathbf{p}_0$)
end if

Subroutine: ExaminePrefix($\mathbf{p}$)
if $L_{\lambda,\min}(\mathbf{p}) < L_\lambda(\mathbf{a})$ **then**
    if $\text{len}(\mathbf{p}) = |S|$ **then**
        $\mathbf{a} = \mathbf{p}_1$
    else
        BranchAndBound($\mathbf{p}$)
    end if
end if

4.3. Complexity considerations

The worst case complexity of the branch and bound technique is $O(2^n)$, since the number of recursive calls is bounded by $O(2^n)$ and the update of the bounds $\epsilon_{\min,j}$ and $\rho_{\min}$ at each recursive call can be carried out in constant time. In practice, however, the pruning of the search space allows for a substantial speed-up of the algorithm. Still, its computational complexity is of a sufficiently high magnitude to preclude its application in practice.

5. GREEDY APPROXIMATION

Here, we approximatively solve (6), in a greedy iterative manner. In particular, at each iteration we choose the best single vertex to be added to the already selected subset A, until the limit on the set size $|A|$ is reached. There are two phases of the algorithm. In the first one, we are concerned with adding nodes to A that will simultaneously contribute to large reductions in hitting time across all graphs of the ensemble. In the second phase, once the mean hitting time on every graph G_j, for $j \neq 1$, has been sufficiently reduced, the algorithm focuses on minimizing the mean hitting time on the social graph, i.e. G_1, exclusively. The second phase of the algorithm takes advantage of the fact that the mean hitting times are monotonically decreasing, on all graphs, as a function of the set size $|A|$. Therefore, even the mean hitting times $\mathbb{E}(T_j), j > 1$, will nonetheless continue to be reduced during the second phase as well.

We start with $A = \emptyset$. For each vertex $i \notin A$, the algorithm computes the reduction in mean hitting time that will be achieved on each graph $j = 1, \ldots, n$, if the vertex i is added to A. That is,

$$\Delta\,\mathbb{E}(T_j)_i = \mathbb{E}(T_j^{\{A \cup i\}}) - \mathbb{E}(T_j^A)\,, \tag{8}$$

where we set $\mathbb{E}(T_j^\emptyset) = 0, \forall j$, on the beginning.

Next, the maximum improvement (reduction) in mean hitting time on each graph is identified with

$$\text{m}\Delta\,\mathbb{E}(T_j) = \min_{i \notin A} \Delta\,\mathbb{E}(T_j)_i\,. \tag{9}$$

Finally, the algorithm computes a compound mean hitting time term for each vertex candidate $i \in S \setminus A$ using

$$\overline{\Delta\,\mathbb{E}(T)}_i = \begin{cases} \prod_{j=1}^n \frac{\Delta\,\mathbb{E}(T_j)_i}{\text{m}\Delta\,\mathbb{E}(T_j)} & : \text{ if } |A| \leq \beta\kappa_1 \text{ and } \mathbb{E}(T_j) > \kappa_j\,, j > 1\,, \\ \Delta\,\mathbb{E}(T_1)_i & : \text{ otherwise}\,. \end{cases} \tag{10}$$

Note that the first and second cases in (10) above signify respectively the first and second phases of the algorithm. The conditions of the first case in (10) control the greediness of the algorithm with respect to meeting the constraints on the hitting times on the graphs G_j, for $j = 2, \ldots, n$. With the parameter $\beta \in (0, 1)$ we can adapt the duration of the two phases of the algorithm in order to enhance its performance. At the same time, by the virtue of considering a product mean hitting time term for its first phase, the algorithm still selects those vertices to be included in A that will simultaneously contribute to large mean hitting time reductions across all graphs. Therefore, this will additionally insure that the algorithm will not dramatically underperform relative to $\mathbb{E}(T_1)$ during this stage. Once the first phase is over, the algorithm exclusively focuses on reducing further the mean hitting time over G_1, as evident from (10).

An operation cycle of the algorithm is completed by selecting the best candidate vertex via the following optimization

$$k = \begin{cases} \arg\min_{i \in S \setminus A} \overline{\Delta\,\mathbb{E}(T)}_i & : \text{ if } A = \emptyset\,, \\ \arg\max_{i \in S \setminus A} \overline{\Delta\,\mathbb{E}(T)}_i & : \text{ otherwise}\,. \end{cases} \tag{11}$$

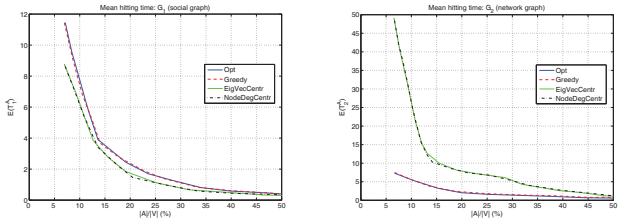
Note that we need to differentiate the two cases $A = \emptyset$ and $A \neq \emptyset$, hence the specific form of Equation (11) above. The algorithm then updates A as $A = \{A \cup k\}$. The procedure described above is repeated until $|A| = \kappa_1$.

The greedy optimization can be solved in a decentralized manner, which makes it additionally appealing for applications where access to the full graphs $G_j, j = 1, \ldots, n$, comprising the multi-graph ensemble is not feasible. In particular, the computational steps of the optimization require the knowledge of the mean hitting time vector τ_e^A associated with the vertex set V, in the case of each graph. The computation of τ_e^A in turn is carried out in a centralized fashion by the matrix inversion in (4), which requires a complete knowledge of each graph G_j. This critical, but computationally intensive, step of the optimization can instead be relegated to the graph's nodes that will compute τ_e^A in a decentralized manner, thereby reducing the overall complexity of the greedy optimization further. Due to space limitations, detailed description cannot be included here.

6. EXPERIMENTAL RESULTS

6.1. Mean hitting time

We compare the performance of the proposed optimization and its greedy approximation, denoted henceforth as *Opt* and *Greedy*, against that of two conventional techniques for node selection in social networks. These are the well-known *centrality* measures that select nodes either according to their vertex degree in the social graph [11] or the value that they exhibit as part of the dominant eigenvector of the graph [12]. Henceforth, we will respectively refer to these two approaches as *NodeDegCentr* and *EigVecCentr*. The mean hitting times on the graphs G_1 and G_2 associated with the four techniques under comparison are shown in Figure 1.



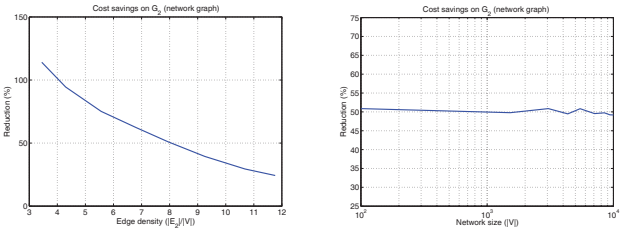
(a) (b)

Fig. 1. Mean hitting time performance on graphs (a) G_1 and (b) G_2.

In particular, it can be seen from Figure 1a that the centrality techniques slightly outperform the proposed optimization methods on the social graph G_1. This is expected, since *Opt* and *Greedy* sacrifice some of the $\mathbb{E}(T_1^A)$ performance in order to keep $\mathbb{E}(T_1^A)$ on G_2 in check. Still, the relative differences in mean hitting time between *NodeDegCentr* and *EigVecCentr* on one hand and *Opt* and *Greedy*, on the other, are rather insignificant for small $|A|$ and tend to diminish even further as the size of A is increased. On the other hand, since *NodeDegCentr* and *EigVecCentr* exclusively focus on the social graph in their operation, we can see from Figure 1b that they exhibit quite long mean hitting times on G_2, relative to our optimization techniques. This can be a substantial shortcoming, as it can lead to extensive sampling (data network) cost and consequently poor performance of the underlying application employing them.

6.2. Data network cost

Here, we consider that the graph G_2 represents the data network connections between the vertices in V and its edge weights correspond to the cost of data communication (transport) on these connection links. That is, G_2 provides an abstraction of the underlying data network serving the online community. The shorter mean hitting time on G_2 that our optimization enables would map to lower network cost when data communication is carried out between the nodes comprising A and the rest of the vertices in V. Figure 2a shows the cost reduction in percent that *Greedy*[3] provides relative to *EigVecCentr* as a function of the edge density of G_2. It can be seen that we observe cost savings on the order of more than 100% for lower edge densities of G_2 that are then gradually reduced as the average number of edges per vertex is increased. This outcome is expected and is due to the fact that the choice of A becomes less critical as the number of edges in a graph is increased. Still, even for very high edge densities our optimization provides non-trivial cost savings, on the order of 30-40%, as evident from Figure 2a.



(a) (b)

Fig. 2. Cost savings (G_2) versus (a) Edge density; (b) Network size.

[3]Here, we only consider *Greedy* as a representative of our optimization framework, due to the sizes of the graphs employed in the experiments.

Next, we examine in Figure 2b the cost reduction relative to the centrality techniques, as a function of the vertex set size $|V|$, for a fixed average edge density of G_2. It can be seen that *Greedy* provides a consistent and practically constant gain of 50% relative to *EigVecCentr* over the whole range of values considered for the size of the online community. The observed improvement in performance is encouraging as it can lead to efficient algorithms for information diffusion that will take advantage of our methodology in a number of important applications spanning recommendations systems, information retrieval, and viral marketing, to name a few examples. It should be noted that the design of prospective multi-graph versions of the reference techniques is beyond the scope of the present paper.

The experiments described thus far have been carried out on synthetic graphs generated according to statistics of social and data network graphs reported in the literature [13]. We also ran the same set of experiments on an actual Facebook data set and observed very similar results that due to space constraints cannot be included here.

7. CONCLUSION

We show that additional layers of graph information associated with an online community should be taken into account when selecting its most influential members. Our approach formalizes for the first time the process of node selection as a minimization of multiple hitting times defined on a multi-graph ensemble. By the virtue of enabling much shorter times to reach the selected influential nodes from any vertex in the social graph, the performance gains that our framework delivers can be interpreted as both data network communication cost savings and increased rate of information dissemination. The latter factor in turn can be additionally interpreted as increased customer satisfaction and profit return, for the operator of the community site.

8. REFERENCES

[1] A. Bagherjeiran and R. Parekh, "Combining behavioral and social network data for online advertising," in *Proc. Int'l Conf. Data Mining Workshops*. Washington, D.C., USA: IEEE, Dec. 2008, pp. 837–846.

[2] R. Bhatt, V. Chaoji, and R. Parekh, "Predicting product adoption in large-scale social networks," in *Proc. 19th Int'l Conf. Information and Knowledge Management*. Toronto, ON, Canada: ACM, Oct. 2010.

[3] C.-Y. Lin, K. Ehrlich, V. Griffiths-Fisher, and C. Desforges, "Smallblue: People mining for expertise search," *IEEE Multimedia*, vol. 15, no. 1, pp. 78–84, Jan.-Mar. 2008.

[4] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, pp. 415–444, Aug. 2001.

[5] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn, "The rise of people-centric sensing," *IEEE Internet Computing*, July-Aug. 2008.

[6] G. Casella and R. L. Berger, *Statistical Inference*, Jun. 2001.

[7] S. Borgatti, "Centrality and network flow," *Social Networks*, vol. 27, no. 1, pp. 55–71, 2005.

[8] E. Atsan and O. Özkasap, "Applicability of eigenvector centrality principle to data replication in MANETs," in *Proc. Symp. Computer and Information Sciences*. Ankara, Turkey: IEEE, Nov. 2007.

[9] X. Shi, M. Bonner, L. A. Adamic, and A. C. Gilbert, "The very small world of the well-connected," in *Proc. 19th Conf. Hypertext and Hypermedia*. Pittsburgh, PA, USA: ACM, Jun. 2008, pp. 61–70.

[10] A. Ravindran, D. T. Phillips, and J. J. Solberg, *Operations Research: Principles and Practice*. John Wiley & Sons, Jan. 1987.

[11] L. C. Freeman, "Centrality in social networks: Conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1979.

[12] P. Bonachich, "Power and centrality: A family of measures," *Social Networks*, vol. 92, no. 5, pp. 1170–1182, 1987.

[13] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 15 Oct. 1999.