Repetition-Aware Content Placement in Navigational Networks

Dóra Erdös
Boston University
Boston, MA
edori@bu.edu

Vatche Ishakian *
Raytheon BBN Technologies
Cambridge, MA
vishakia@bbn.com

Azer Bestavros
Boston University
Boston, MA
best@bu.edu

Evimaria Terzi
Boston University
Boston, MA
evimari@bu.edu

ABSTRACT

Arguably, the most effective technique to ensure wide adoption of a concept (or product) is by repeatedly exposing individuals to messages that reinforce the concept (or promote the product). Recognizing the role of repeated exposure to a message, in this paper we propose a novel framework for the effective placement of content: Given the navigational patterns of users in a network, *e.g.,* web graph, hyperlinked corpus, or road network, and given a model of the relationship between content-adoption and frequency of exposition, we define the *repetition-aware content-placement* (RACP) problem as that of identifying the set of B nodes on which content should be placed so that the expected number of users adopting that content is maximized. The key contribution of our work is the introduction of *memory* into the navigation process, by making user conversion dependent on the number of her exposures to that content. This dependency is captured using a conversion model that is general enough to capture arbitrary dependencies. Our solution to this general problem builds upon the notion of absorbing random walks, which we extend appropriately in order to address the technicalities of our definitions. Although we show the RACP problem to be NP-hard, we propose a general and efficient algorithmic solution. Our experimental results demonstrate the efficacy and the efficiency of our methods in multiple real-world datasets obtained from different application domains.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*

Keywords

Optimization, Markov chains, Navigational networks

*This work was completed while the author was a PhD student at Boston University

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

1. INTRODUCTION

Motivation: There is ample evidence in the literature that the probability of internalizing a concept or buying a product (user conversion) is dependent on the number of times that an individual user is exposed to information or advertisement related to that concept or product. As Artistotle put it *"it is frequent repetition that produces a natural tendency".* In education, repetition is recognized as an effective pedagogical tool; repetition deepens and hastens students' engagement and understanding processes [5, 22]. In marketing, repeated exposure to a product is key to the success of marketing campaigns [17]. In politics, repeating specific messages in stump speeches or in mass media advertisements is effective in influencing public opinion, and thus critical to the success of political campaigns [1, 21]. The effects of repetition are not always positive: while repeated exposure to a message increases one's ability to internalize a concept in an educational setting, it may yield undesirable outcomes in a different setting – for example, the probability of purchasing a product decreases dramatically with repeated exposure (more than twice) of the product to a customer [1].

Problem: Motivated by the role that repeated exposure to content plays in these various settings, in this paper we define and study the *repetition-aware content-placement* (RACP) problem in *navigational networks*, *i.e.,* networks in which the (directed) edges represent the potential of users to transition from one node to another as they navigate through the network. Broadly speaking, given the navigational patterns of users in such a network, and given the relationship between the level of user exposure to content and the probability of user conversion, the RACP problem is that of identifying the set of k nodes on which content should be placed so that the expected number of users adopting the content (*i.e.,* the conversion rate) is maximized.

Applications: Instances of RACP occur in multiple domains. For example, consider the problem of superimposing content on the navigational network defined by the hyperlink structure of the web. Here, the challenge is the identification of the set of k web pages (nodes of the navigational graph) on which content should be placed in order to maximize the impact of an advertisement campaign, *e.g.,* placement of slogans that raise awareness about a social issue, or placement of advertisement about a product or a political party. As another example, consider the problem of providing recommendations for additional content to readers of on-line

[1] `http://www.mediabizbloggers.com/bill-harvey/50150992.html`

corpora. Here, the reader (a student) navigates a body of knowledge (an on-line textbook) – not necessarily serially – by following links that underscore dependencies between units of the corpus (e.g., sections and chapters), and the challenge is to identify the best set of units where additional content (further readings, references, exercises) could be linked or recommended so as to maximize the probability of access to such additional content. The RACP problem is applicable to offline physical navigational networks as well – the canonical example being road networks. Here users navigate a set of interconnected locations, and the challenge is the placement of billboard advertisements at the right locations, so as to maximize the impact on travelers/commuters.

Model: The main components of our setting are the user's *navigational* and *conversion* models. The navigational model assumes that user's navigation through the nodes of the network is modeled by a Markov chain, *i.e.,* a random walk over the navigational graph. The conversion model specifies the probability of a user adopting content as a function of the number of times that content is shown to the user within a single walk over the navigational network. Our incorporation of a conversion model fundamentally changes the nature of content placement by making user conversion dependent on the number of times that content is shown to the user. Said differently, the novelty of our work is the introduction of *memory* into the navigation process. In our work, rather than focusing on a particular type of relationship between number of views and probability of content adoption, we consider generic conversion models that handle arbitrary dependencies between the number of views and the probability of content adoption.

Contributions: Problems of picking important (or target) nodes in networks have been studied extensively in the past, both in information-flow (or social) networks [7, 9, 11, 16], in transportation [2, 3, 13, 14, 20] and in navigational networks [6, 8]. A thorough examination of this related work is presented in Section 2. The key difference between that prior work and ours is that the former assumes *memoryless* navigation or information flow processes, and as a result has an implicit conversion model that presumes the independence of number of views and conversion probability. The explicit modeling of this dependence and the flexibility of our model to capture arbitrary forms of this dependency makes our work more general in the sense that prior work represents a subset of the scenarios that are possible to consider using our approach. Another salient feature of our work is that our formulation and algorithmic treatment of the RACP problem are general in the sense that they apply to *any* conversion model as long as the conversion probability can be expressed as a function of the number of times content is presented to the user in a single random walk over the navigation graph. The incorporation of an arbitrary conversion model (memory) into a navigation (random walk) process raises new modeling as well as computational challenges, which comprise the main technical contributions of our paper. We address this new class of problems by building upon the notion of random walks with absorbing states. More specifically, we propose solutions to our RACP problem based on analysis using absorbing random walks, and we demonstrate that our techniques are scalable and thus useful for real-world data-analysis tasks. To the best of our knowledge we are the first to leverage such techniques not only in the context of content placement in navigational networks, but also in the context of node-selection in general.

2. RELATED WORK

Although, to the best of our knowledge, we are the first to propose and solve the RACP problem, which encompasses both a navigational and a *general* conversion model, our work is related to a large body of existing work on information, transportation and online navigational networks.

Information networks: Recently, there has been a lot of work in the computer-science literature on identifying a set of nodes of a social network to advertise a product or to immunize so that the spread of the product or the spread of an epidemic in the network is maximized or minimized. Different assumptions about how information items propagate in the network has led to a rich literature of node-selection methods [7, 9, 11, 16, 18, 19]. The key difference between our work and all these methods is that the latter assume that the underlying network is an information network on which items (rather than users) propagate. As a result, the models adopted in such existing work are information-propagation models and thus, cannot be used to model the navigation of users in a navigational network. Hence, despite the fact that at a high level we also need to pick a subset of the nodes of our network, our objective – i.e., the maximization of users' conversion rate – does not have an analogue in the social-network literature. More specifically, the models we propose here are specific to navigational networks and lead to problem definitions that require the development of new machinery in order to be solved.

Transportation networks: Related to ours is the work on the *flow-capturing location allocation problem* (FCLA) in transportation networks. This problem has originally been introduced in 1990 by Hodgson [13]. In the FCLA problem, the input consists of a network as well as the customer traversals in the form of flows between source and destination pairs. The objective is to locate facilities in a set of k nodes so as to maximize the number of customers who encounter at least one facility in their flow through the network. In his original paper, Hodgson proves that this version of the location-allocation problem is NP-hard, but empirically shows that a greedy algorithm can be quite efficient in solving it. The problem we study here is different from this original version of the FCLA problem in three ways: First, while in FCLA the repetition of interceptions (i.e., multiple encounters of customers with facilities) is ignored, our work focuses on optimizing the interception of the users' navigation given the impact that repetitions have on the users' tendency to convert. Second, Hodgson assumes that the interception of a flow is a deterministic process, i.e., a flow is either intercepted or not, hence set-cover type of reasoning works for his approach. In our case the paths of users is only intercepted in a probabilistic sense. Third, in our work we assume the navigation model is Markovian, while in FCLA the navigation paths of users from sources to destinations are a priori known and deterministically defined. As a result, both our model as well as the computational challenges we need to resolve are different from those that arise in the FCLA problem.

Even in the domain of transportation networks, the assumption that all source-destination flows are known a priori proved impractical. As a result, there exists work on

variants of the original FCLA problem where partial flow information is assumed [2, 3]. In these cases, the available navigation information specifies the fraction of flows that pass through every node and its neighbors. Despite the fact that this navigation model resembles ours, existing work still ignores the effect of repetitive interceptions on users. As a result, the underlying combinatorial problems that appear in existing work [2, 3] are different from ours. After all, the algorithms used for solving these existing variants of FCLA are based on non-linear integer programming and total-reward Markov decision processes, while our solution is based on deploying Markov chains with absorbing states.

Navigational networks: More recent work on node - selection in online navigational networks includes the work of Chierichetti *et al.* [8] and Charikar *et al.* [6]. In their work Charikar *et al.* [6] assume a similar propagation model and objective to ours, in that users traverse a Markov chain and the ultimate goal is to assign content to certain states in this chain. Despite the similar objective function, the underlying problem they solve is completely different from ours; in their setup users can either be in a targeted or non-targeted population and the goal is to intercept the largest possible fraction of the targeted population. In their setting, interception of users' is deterministic and repetition does not play any role. As opposed to this, in our work users are only intercepted with some probability in chosen states and this probability may change with the number of times an interception happens.

Chierichetti *et al.* [8] assume a user-navigation model similar to ours and their goal is to find an optimal placement of online advertisements. In their setting, ads are placed to the nodes of the navigation network and there is a utility assigned to a user seeing an ad, that depends both on the state and the specific ad that is shown. When an ad is shown the user may stop with some probability (go to an exit state of the chain) or continue traversing. While the user propagation model of Chierichetti *et al.* is similar to ours, their conversion model is different. In special, they do not distinguish between the first and subsequent views of the same ad. They propose a very elegant LP solution for their problem. However, the method they derive is inadequate to solve the RACP problem. Due to the fact that in our setting the conversion probabilities are affected by repeated traversal through the same state, the size of the corresponding LP program in their solution would blow up.

3. PROBLEM DEFINITION

The input to our problem consists of the *navigational* and the *conversion* model. The navigational model is a network; the nodes of this network correspond to entities; the user navigates through the entities by following the links of the underlying graph (directed or undirected) based on probabilities associated with the links. The conversion model quantifies the relationship between the number of times a user views a particular content and the probability of her adopting the content (or converting to the content). The objective of our work is to place content in the network at locations so as to maximize the probability of conversion.

Throughout the paper, we assume as input a navigational graph $G = (\mathbf{S}, E)$, which is a directed or undirected graph with nodes $\mathbf{S}$, edges E and $|\mathbf{S}| = n$.

Navigational model: We assume that the users' traversal of the network follows the traditional Markov model; users navigate between a set of states in a randomized way, transitioning between states with given probabilities, such that the transition probabilities define a Markov chain on this set of states. Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{P} \rangle$ denote this Markov chain, set $\mathbf{S}$ contains n states $\mathbf{S} = \{s_1, s_2, \ldots, s_n\}$ and $\mathbf{P}$ contains the transition probabilities $\mathbf{P}(i, j)$ of a user moving from state s_i to state s_j. We think of $\mathbf{P}$ as an $n \times n$ matrix, that is the *transition matrix* of $\mathcal{M}$. Note that the state space $\mathbf{S}$ of the Markov chain is simply the set of nodes of the navigational graph $G = (\mathbf{S}, E)$.

For the rest of the discussion, we will assume that $\mathcal{M}$ is irreducible and aperiodic, and thus has a stationary distribution π[2]. Finally, in order to make our model more realistic, we assume that there is an upper bound on the maximum number of hops the user is taking, i.e., this bound encodes that the user quits his navigational session in finite time. We denote this bound by $M_{\max}$.

Conversion model: This model depicts the user's behavior upon being exposed to some content in one of the states. Upon viewing content $\mathbf{c}$ the user has two possible actions: either *convert* (i.e., click on the link, adopt some view or buy the advertised product) and quit the traversal of the network or continue without taking any action. The conversion model provides the probability with which the user chooses in any given situation between these two options. Note, that a user may only quit the navigation by either converting or by exceeding the maximum number of hops.

In this paper, we focus on memory-full conversion models, i.e., conversion models for which the probability of a user converting to the content $\mathbf{c}$ depends on the *number of times* she has been exposed to this content before – while it does not depend on the specific path that the user has followed. To describe this model in more detail we introduce the notion of *levels*.

DEFINITION 1. *We say that a user is in level ℓ if she has been exposed to content $\mathbf{c}$ exactly ℓ times.*

A result of the above definition is that the user is in level 0 when she has not seen the content yet. That is, every user starts her traversal of the network in level 0. When a user, who is at level ℓ is exposed to the content but decides not to convert, she moves to level $(\ell + 1)$.

The probability that a user converts to content $\mathbf{c}$, when presented with it in state s, depends on the state itself as well as on the user's level ℓ, but not on the user's path during her navigation. We denote this probability by $C(\ell, s)$. Naturally, the probability that a user continues the traversal – without converting – is $(1 - C(\ell, s))$. We call $C(\ell, s)$ the *conversion probability* in state s at level ℓ.

Note that the probability $C(\ell, s)$ can be any arbitrary function of s and ℓ. For example, if the number of repetitions of the content's viewings increases (resp. decreases) the probability of adoption, then we will assume that $C(\ell, s)$ is monotonically increasing (resp. decreasing) with ℓ. However, such monotonic relationship is not necessary for our framework. Our intuition is that the probability of the user's conversion increases for the first couple of exposures to the content and then it decreases.

[2]In fact, our method only makes use of the irreducibility property of Markov chains. In our experiments we compute the PageRank with a dampening factor [4] of states instead of the stationary distribution.

The only assumption we make with respect to the dependency of $C(\ell, s)$ on ℓ is that there is a sufficiently large number K, such that if the user did not convert after level K, then the probability of conversion becomes infinitesimal small. That is, $C(\ell, s) \to 0$ for $\ell \geq K$ and for *any* s. This assumption also ensures that the number of different conversion probabilities per state is at most K.

The dependency of $C(\ell, s)$ on the state s may also be arbitrary. In some cases, the conversion probability $C(\ell, s)$ may depend on the relevance of the content $\mathbf{c}$, which we are placing to the node s, and the content of the node s itself. For example, an ad's placement on a webpage depends on the topic of the page. In other cases, the conversion probability $C(\ell, s)$ may only depend on factors irrelevant to the content or the node.

Expected conversion rate: The objective in the RACP problem is to maximize the expected probability of a user converting to content $\mathbf{c}$, given the navigational and conversion models. We are now ready to give the formal definition of this objective function.

For this, let $\mathcal{M}\langle \mathbf{S}, \mathbf{P} \rangle$ be the user's navigational model. Moreover, if $L = \{1, 2, \ldots, K\}$ is the set of all possible levels of a user then let $C : L \times \mathbf{S} \to [0, 1]$ denote the input conversion model. Finally, let $R(\ell, s)$ denote the probability that the user is encountering content $\mathbf{c}$ for the ℓ-th time when she is visiting state s. Note that since the user has not yet quit the navigation she has neither converted to $\mathbf{c}$, nor has she reached the maximum number of steps $M_{\max}$. Although the definition of $R(\ell, s)$ is conceptually easy, computing the value of $R(\ell, s)$ is computationally challenging. In order to maintain the smoothness of the flow of the paper, we assume for now that $R(\ell, s)$ can be computed and we give the details of this computation in Section 4.2.

If copies of content $\mathbf{c}$ are placed on a subset of states $\mathbf{A} \subseteq \mathbf{S}$, then the expected *conversion rate* of content placement $\mathbf{A}$ is given by the formula:

$$\texttt{CR}(\mathbf{A}) = \sum_{s \in \mathbf{A}} \sum_{\ell=0}^{K} R(\ell, s) C(\ell, s). \quad (1)$$

Note that Equation (1) is the probability that a user reaches a state $s \in \mathbf{A}$ and converts to content $\mathbf{c}$, after having encountered content $\mathbf{c}$ for ℓ times in the past, where $\ell \in L$.

Since our goal is to actually find the content placement $\mathbf{A}$ for which Equation (1) is maximized, we define the RACP problem as an optimization problem as follows:

PROBLEM 1 (RACP). *Given the navigational model $\mathcal{M} = \langle \mathbf{S}, \mathbf{P} \rangle$, the conversion model $C(\ell, s)$ for every $\ell \in L$ and $s \in \mathbf{S}$, and a budget B, assign content to at most B states $\mathbf{A} \subseteq \mathbf{S}$ so that the expected conversion rate $\texttt{CR}(\mathbf{A})$ is maximized.*

THEOREM 1. *The* RACP *problem is NP-hard.*

PROOF. We prove the theorem by reducing the decision version of the VERTEX COVER problem [12] to the decision version of the RACP problem.

The decision version of the VERTEX COVER problem takes as input a graph $G' = (V', E')$ and asks whether there exists a set of vertices $U \subseteq V'$ of size at most k such that every edge in E' is incident to at least one of the vertices in U.

We transform the above instance of VERTEX COVER to an instance of the decision version of our problem as follows. We assume that our navigation graph $G = (\mathbf{S}, E)$ has the same nodes as G' (i.e., $\mathbf{S} = V$) and the same set of edges as E' (the undirected edges in E' become bidirectional edges in E). Our navigational model is a simple Markov chain defined on G. Further, we define our conversion model as follows: for any state $s \in \mathbf{S}$, the user converts to content $\mathbf{c}$ the first time she encounters $\mathbf{c}$. That is, for every $s \in \mathbf{S}$ we have that $C(0, s) = 1$. We can now show that there exists a vertex cover of size k in G' iff in the above instance of the RACP problem there exists a set of k nodes $\mathbf{A} \subseteq \mathbf{S}$ such that for $M_{\max} = 1$ the expected conversion rate for $\mathbf{A}$ is equal to 1, i.e., $\texttt{CR}(\mathbf{A}) = 1$. □

4. SOLVING THE RACP PROBLEM

In this section, we give a greedy algorithm for solving the RACP problem. We also identify the connection of our problem to random walks with absorbing states and demonstrate how this connection is exploited within the implementation of our algorithm.

4.1 The `Greedy` algorithm

Our greedy algorithm, which we call `Greedy`, forms solution $\mathbf{A}$ iteratively; at each iteration it adds the node that causes (locally) the most increase in the objective function.

More specifically, `Greedy` starts from an empty set $\mathbf{A} = \emptyset$. At iteration i, a new state s is added to $\mathbf{A}$, such that $\texttt{CR}(\mathbf{A} \cup \{s\})$ is maximized. The algorithm terminates when either the budget B of states for content placement is exceeded, or there is no state that increases the expected conversion rate. In many applications, not all states are available for content placement. For this reason, we keep a set of candidate states $\mathcal{S} \subseteq \mathbf{S}$ and only consider states in $\mathcal{S}$ to add to $\mathbf{A}$. The pseudocode of the `Greedy` algorithm is given in Algorithm 1.

Algorithm 1 The `Greedy` alorithm for the RACP problem.

Input: Markov chain $\mathcal{M}\langle \mathbf{S}, \mathbf{P} \rangle$, budget B, candidate list $\mathcal{S}$ and conversion rates $C(\ell, s)$ for every $\ell \in \{0, \ldots, K\}$ and every $s \in \mathbf{S}$.
Output: set of states $\mathbf{A}$ and $\texttt{CR}(\mathbf{A})$.
$\mathbf{A} \leftarrow 0$
$\texttt{CR}(\mathbf{A}) \leftarrow 0$
for $i = 1 \ldots B$ **do**
  $s = \text{argmax}_{s' \in \mathcal{S}} \texttt{CR}(\mathbf{A} \cup \{s'\})$
$\mathbf{A} = \mathbf{A} \cup \{s\}$
$\mathcal{S} = \mathcal{S} \setminus \{s\}$

In terms of running time, the most computationally expensive step of `Greedy` is the computations done inside the for loop, i.e., computing $\texttt{CR}(\mathbf{A} \cup \{s'\})$ (line 4 of Algorithm 1). If the time required for computing function $\texttt{CR}()$ is T, then the running time of `Greedy` is $O(B|\mathcal{S}|T)$, which in the worst case (i.e., when $\mathcal{S} = \mathbf{S}$) is $O(BnT)$.

Note that at each iteration of the `for` loop in line 4 all candidates need to be evaluated in order to choose the one with the largest marginal benefit. Although this can be time consuming, we observe that the evaluation of each candidate can be done independently of the rest. Therefore, we have implemented a parallel version of `Greedy`, which we call `Par-Greedy`, and which evaluates each candidate separately. Thus `Par-Greedy` is $O(n/q)$ times faster than the serial version of `Greedy` shown in Algorithm 1, where q depends on the number of cores of the underlying hardware.

The details of how we evaluate $\texttt{CR}(\mathbf{A})$ for any $\mathbf{A} \subseteq \mathbf{S}$ are given in the next paragraph.

4.2 Computing the expected conversion rate

The computation of $\texttt{CR}(\mathbf{A})$ for any $\mathbf{A}$ requires the evaluation of Equation (1). Since $C(\ell, s)$ for any level ℓ and any state s is provided as part of the input (i.e., the conversion model), the main challenge in the evaluation of Equation (1) stems from computing the values of $R(\ell, s)$ for every $\ell \in \{0, \ldots, K\}$ and every $s \in \mathbf{S}$. Next, we show how these computations can be done using the notion of absorbing Markov chains [10, 15].

Absorbing Markov chains: Given an underlying graph $H = (\mathbf{X}, Q)$, consisting of nodes X and edges Q, an *absorbing Markov chain* $\mathcal{C} = \langle \mathbf{X}, \mathbf{Q} \rangle$ defines an absorbing random walk on H. The statespace of this walk is $\mathbf{X}$ (i.e., the nodes of H) and there are two types of states in $\mathbf{X}$: *absorbing* and *transient*. A state $x \in \mathbf{X}$ is *absorbing* if the random walk transitions into this node, but not out of it (and thus, the random walk is absorved in state x). Let $\mathbf{B} \subseteq \mathbf{X}$ denote the set of absorbing states. The remaining states $\mathbf{U} = \mathbf{X} \setminus \mathbf{B}$ define the set of non-absorbing or *transient* states.

Given this partition of the states, the transition matrix of this random walk can be written as follows:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q_{UB}} & \mathbf{Q_{UU}} \\ \mathbf{I} & \mathbf{0} \end{pmatrix}. \tag{2}$$

If $|X| = N$, then in the above equation, $\mathbf{I}$ is an $(N - |\mathbf{U}|) \times (N - |\mathbf{U}|)$ identity matrix and $\mathbf{0}$ a matrix with all its entries equal to 0; $\mathbf{Q_{UU}}$ is the $|\mathbf{U}| \times |\mathbf{U}|$ sub-matrix of $\mathbf{Q}$ with the transition probabilities between transient states; and $\mathbf{Q_{UB}}$ is the $|\mathbf{U}| \times |\mathbf{B}|$ sub-matrix of $\mathbf{Q}$ with the transition probabilities from transient to absorbing states.

An important quantity of an absorbing random walk is the expected number of visits to a transient state y when starting from a transient state x, before being absorbed. The probability of transitioning from x to y in exactly ℓ steps is the (x, y) entry of the matrix $\mathbf{Q}^{\ell}_{\mathbf{UU}}$. Finally, the matrix

$$\mathbf{Q_{UB}} = \mathbf{Q}^{\ell}_{\mathbf{UU}} \mathbf{Q_{UB}} \tag{3}$$

is an $|\mathbf{U}| \times |\mathbf{B}|$ matrix, with $\mathbf{Q_{UB}}(x, y)$ being the probability that a random walk which starts at a transient state x ends up being absorbed at state $y \in \mathbf{B}$.

Absorbing Markov chains and the RACP **problem:** In order to illustrate how absorbing random walks can be leveraged by our problem, let us consider the navigational graph $G = (\mathbf{S}, E)$ and the corresponding navigational model $\mathcal{M} = \langle \mathbf{S}, \mathbf{P} \rangle$ and conversion model $C(\ell, s)$ for $\ell \in \{0, \ldots, K\}$ and $s \in \mathbf{S}$. Further assume that a subset of states $\mathbf{A} \subseteq \mathbf{S}$ have been selected for placing content $\mathbf{c}$.

Given the above we will define the *extended navigational graph* $\widehat{G} = (\widehat{\mathbf{S}}, \widehat{E})$ as follows: for every node $s \in \mathbf{S} \setminus \mathbf{A}$ there exist a node $s' \in \widehat{\mathbf{S}}$, and for every node $s \in \mathbf{A}$ we create two copies of it in $\widehat{\mathbf{S}}$ – one denoted by s_t and the other denoted by s_b. We call s_t the *transient copy of* s and s_b the *absorbing copy of* s. For the rest of the discussion, we will use $\mathbf{A}_t$ and $\mathbf{A}_b$ to denote the set of all transient copies and the set of all absorbing copies appearing in $\widehat{\mathbf{S}}$ due to states $s \in \mathbf{A}$. Clearly, $|\mathbf{A}_t| = |\mathbf{A}_b|$ and $|\widehat{\mathbf{S}}| = |\mathbf{S}| + |\mathbf{A}|$. The edges among the nodes in $\mathbf{S} \setminus \mathbf{A}$ are the same both in G and $\widehat{G}$. Finally, the transient copy s_t of $s \in \mathbf{A}$ maintains all the outgoing edges of node s, while the absorbing copy s_b of s maintains all the incoming edges of s.

Given $\widehat{G}$, we also define the absorbing Markov chain $\widehat{\mathcal{M}} = \langle \widehat{\mathbf{S}}, \widehat{\mathbf{P}} \rangle$. In this Markov chain, the nodes $\mathbf{A}_b$ are absorbing and all other nodes in $\widehat{\mathbf{S}}$ are transient. The transition matrix $\widehat{\mathbf{P}}$ of such an absorbing random walk is defined as follows: the transition probability from a transient state s to an arbitrary state $s' \in \widehat{\mathbf{S}}$ is identical to that in $\mathcal{M}$, thus $\widehat{\mathbf{P}}(s, s') = \mathbf{P}(s, s')$. For absorbing state $s \in \mathbf{A}_b$ the probability of transitioning to any other state $s' \in \widehat{\mathbf{S}}$ is zero and thus, $\widehat{\mathbf{P}}(s, s') = 0$. To make $\widehat{\mathbf{P}}$ a proper transition matrix (i.e., ensure that all its rows sum up to 1) we set $\widehat{\mathbf{P}}(s, s) = 1$ for all absorbing states $s \in \mathbf{A}_b$. Note that matrix $\widehat{\mathbf{P}}$ has the same structure as matrix $\mathbf{Q}$ described in Equation (2).

A transformation from the original Markov chain $\mathcal{M}$ to the absorbing Markov chain $\widehat{\mathcal{M}}$ is shown in Figures 1 and 2. The former figure shows the original Markov chain which corresponds to the navigational graph $G = (\mathbf{S}, E)$. The highlighted nodes in the figure correspond to the set $\mathbf{A}$ on which content $\mathbf{c}$ is placed. The highlighted graph of Figure 2 shows the absorbing Markov chain $\widehat{\mathcal{M}} = \langle \widehat{\mathbf{S}}, \widehat{\mathbf{P}} \rangle$, which corresponds to graph $\widehat{G} = (\widehat{\mathbf{S}}, \widehat{E})$. The highlighted nodes here are the absorbing states of the chain.

Note that the extended graph $\widehat{G}$ and the corresponding absorbing Markov chain $\widehat{\mathcal{M}}$ capture the navigational journey of a user at a single level, e.g., level ℓ. At this level, the user navigates according to her navigational model and once it encounters content $\mathbf{c}$ placed in one of the level's absorbing nodes, the random walk of the user gets absorbed.

In order to capture the journey of the user across levels holistically, we need to create one copy of $\widehat{\mathcal{M}}$ at every level $\ell = \{0, \ldots, K\}$; we denote such copies by $\widehat{\mathcal{M}}^{\ell}$. Now, when the user gets absorbed at s_b at level ℓ (i.e., while she was at random walk $\widehat{\mathcal{M}}^{\ell}$), she directly enters random walk $\widehat{\mathcal{M}}^{(\ell+1)}$; the starting point of this random walk is node s_t. We call this set of connected absorbing Markov chains a *sequence of absorbing Markov chains*. The transition of the user from $\widehat{\mathcal{M}}^{\ell}$ to $\widehat{\mathcal{M}}^{(\ell+1)}$ is shown in Figure 2 and is captured by the dotted arrows that connect nodes from different levels.

In practice, we never actually construct this sequence of absorbing Markov chains, neither do we need to do any computation on the sequence itself. However, the above description provides a nice intuition and a conceptual understanding of the computations that follow.

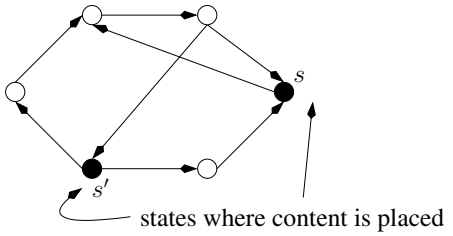


Figure 1: Markov chain $\mathcal{M} = \langle \mathbf{S}, \mathbf{P} \rangle$, with $\mathbf{A} = \{s, s'\}$ picked as states on which content $\mathbf{c}$ is placed.

Computing CR**:** In this paragraph we compute $\texttt{CR}(\mathbf{A})$ as given in Equation (1). Observe, that as $C(\ell, s)$ is part of the

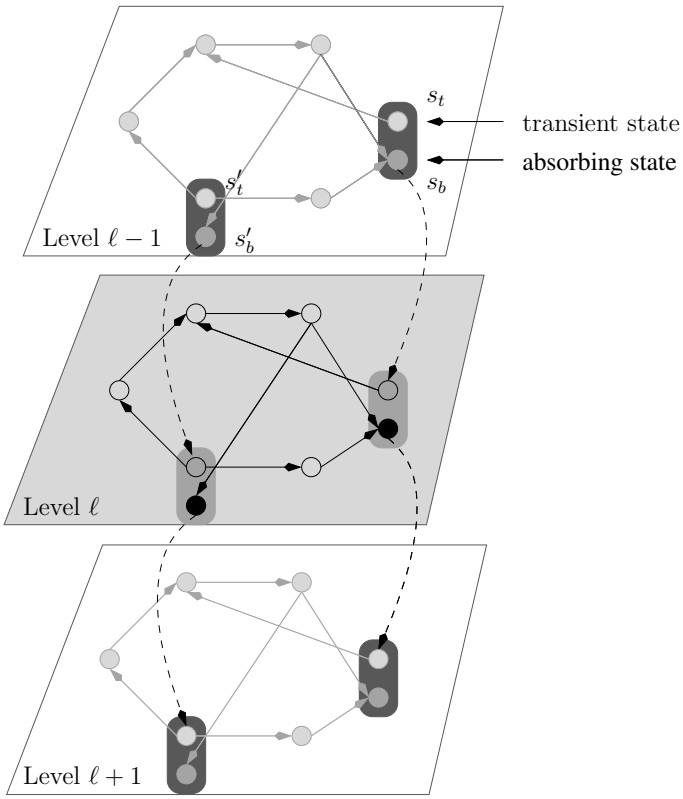


Figure 2: A sequence of absorbing Markov chains $\widehat{\mathcal{M}}^\ell = \langle \widehat{\mathbf{S}}, \widehat{\mathbf{P}} \rangle$. For each state $s \in \mathbf{A}$ of Markov chain $\mathcal{M}$ shown in Figure 1 two states are created: s_b (absorbing) and s_t (transient). The transient and the absorbing copies of the same state are drawn as superstates.

input, we only need to compute $R(\ell, s)$ where s is one of the states in $\mathbf{A}$.

Given the above discussion though, $R(\ell, s)$ is identical to the probability of the user being absorbed in the absorbing copy of s, i.e., node s_b, in $\widehat{\mathcal{M}}^\ell$. Since the user only enters $\widehat{\mathcal{M}}^\ell$ from transient states $t \in \mathbf{A}_t$ then $R(\ell, s)$ can be computed as the sum over all states in $t \in \mathbf{A}_t$ of probabilistic paths of length *at most* $M_{\max}$ ending in s_b and starting at all possible entry points $t \in \mathbf{A}_t$.

The probability of a path of length of *exactly* i between any two states s_1 and s_2 can be computed as the appropriate cell in the i-th power of the transition matrix as $Pr_i(s_1, s_2) = \widehat{\mathbf{P}}^i(s_1, s_2)$. Hence, the probability of a path of length at most $M_{\max}$ from any state $t \in \mathbf{A}_t$ to state s_b is

$$Pr(t, s_b) = \sum_{i=0}^{M_{\max}} \widehat{\mathbf{P}}^i(t, s_b). \quad (4)$$

Speedup: The computation of Equation (4) can be computationally demanding. After all, it requires evaluating the $M_{\max}$-th power of the matrix $\widehat{\mathbf{P}}$. However, since we only need to know the absorption probabilities of the states in $\mathbf{A}_b \subseteq \widehat{\mathbf{S}}$, we can obtain a significant speedup as follows: first we define an auxiliary matrix F of size $|\widehat{\mathbf{S}}| \times |\mathbf{A}_b|$. The columns of this matrix correspond to the absorbing states $\mathbf{A}_b$ of chain $\widehat{\mathcal{M}}$ while the rows of the matrix correspond to all states $\widehat{\mathbf{S}}$ of $\mathcal{M}$. Each column of F has one non-zero element: for $s_b \in \mathbf{A}_t$ we set $F(s_t, s_b) = 1$. That is, there is an 1 in a cell of F if the row and the column of the cell correspond to the transient and the absorbing copies of the same state.

Observe now that $\widehat{\mathbf{P}} \cdot F$ is of the same size as F, and contains at cell (s', s_b) the probability that a random walk starting in state s' will be absorbed in one step in state s_b. Moreover, if we set $F_0 = F$ and apply the recursion

$$F_i = \mathbf{P}_\mathbf{A} \cdot F_{i-1} \quad (5)$$

we get that $F_i(s', s_b)$ is the same as $\mathbf{P}^i_\mathbf{A}(s, a)$; i.e., stores the probability that a random walk that starts at s' gets absorb at s_b. From the theory of random walks with absorbing states, we can observe that matrix F and Recursion (5) allows us to compute the analogue of matrix $\mathbf{Q_{UB}}$ (described in Equation (3)) for the absorbing Markov chain $\widehat{\mathcal{M}}$.

Despite the fact that Equation (5) still involves a matrix multiplication, it is much more efficient to compute in practice, when compared to Equation (4). This is not only because F is of much smaller size than $\widehat{\mathbf{P}}$, but also because the sparsity of $\widehat{\mathbf{P}}$ is maintained and thus the running time of the computation only depends on the number of non-zero entries of $\widehat{\mathbf{P}}$ (i.e., the number of edges $|E|$ of the input navigational graph). Thus, for an absorbing state s_b and transient state t we can compute the probability of a user being absorbed in that state after at most $M_{\max}$ steps by

$$Pr(t, s_b) = \sum_{i=0}^{M_{max}} F_i(t, s_b). \quad (6)$$

Using the above machinery, we can to compute the probabilities $R(\ell, s)$. Observe that $R(\ell, s)$ depends on two things: the probability that the user will end up at level ℓ, and the probability that the user will be absorbed in the absorbing copy of s, i.e., s_b, at level ℓ. Thus $R(\ell, s)$ for every level $\ell = 1 \ldots K$ can be computed by the recursion:

$$R(\ell, s) = \sum_{s' \in \mathbf{A}} R(\ell - 1, s')(1 - C(\ell - 1, s'))Pr(s'_t, s_b) \quad (7)$$

Level 0 is slightly different since at this level the user can start his random walk in the Markov chain in any state. Assuming that the probability of starting at any state is proportional to the state's stationary probability we have that

$$R(0, s) = \sum_{s' \in \mathbf{S}} \pi(s')Pr(s', s). \quad (8)$$

Since $C(\ell, s)$ is a-priori given as an input, we can now compute $\texttt{CR}(A)$ using its definition given in Equation (1).

4.3 Running times

Running time of computing $\texttt{CR}(\mathbf{A})$: Regardless of the maximum number of levels, we only need to compute the absorption probabilities in Equation (6) once, since they are identical in every level. Evaluating this takes as much time as multiplying $\widehat{\mathbf{P}}$ with F, $M_{\max}$ times. Since $|\mathbf{A}| \leq B$ and K and B are constants, then the total running-time for computing $\texttt{CR}(\mathbf{A})$ only depends on the non-zero entries of $\widehat{\mathbf{P}}$ and is thus $O(|E|)$; in practice $|E| \ll n^2$ and thus, the time required for this computation is sub-quadratic.

Running the `Greedy` algorithm: Having described the underpinnings of the computation of $\texttt{CR}(\mathbf{A})$, we can now

describe the details of the Greedy algorithm, shown in Algorithm 1. The algorithm starts from an empty set $\mathbf{A}$ and, given constant integer budget B, it runs for B iterations. At every iteration, a new element is added to $\mathbf{A}$ such that $\texttt{CR}(\mathbf{A} \cup \{s\})$ is maximized. This maximization step is achieved by computing $\texttt{CR}(\mathbf{A} \cup \{s\})$ for every candidate state s and choosing the one which gives the highest CR. If all nodes in $\mathbf{S}$ are considered as candidates for content placement, then CR is computed nB times in Greedy. Plugging in the running time of computing CR, this yields a total running time of $O(n|E|)$. Again, the parallel version of the Greedy algorithm can attain lower running times. The degree of speedup depends on the degree of parallelism allowed by the hardware.

5. EXPERIMENTAL EVALUATION

In this section, we give an experimental evaluation of the Greedy algorithm on real datasets. All our implementations are in Matlab and we conducted our experiments using a 12 CPU cores (Intel Xeon E5-2680 processors, operating at 2.7 GHz) machine with 256GB of 1333 MHz DDR3 RAM.

5.1 Experimental Setup

Datasets: In our experiments, we use real-world datasets. For our experiments with real navigational graphs, we pick graphs that come from domains where the RACP problem is applicable. Further, the choice of datasets is such that they help us demonstrate the scalability of our algorithms. We describe the characteristics of the datasets we use below.

The ROAD dataset: This is the road network of the state of Minnesota[3] The links in the network are undirected and correspond to roads in Minnesota, while nodes correspond to road intersections. The dataset contains 2642 nodes, and 6600 edges. Since this network depicts actual roads it is not only very sparse but the node degree distribution is also quite homogeneous, with degrees ranging from 1 to 10, but most degrees being at most 5. Content placement in this setting can correspond to placing billboards along the roads.

The WEB dataset: This dataset contains the hyperlink structure of the stanford.edu domain in 2001[4]. Nodes of the graph correspond to web pages and the directed edges correspond to hyperlinks between them. The graph contains $10K$ nodes and $36K$ edges. The degree distribution of the graph is power-law with degrees ranging from 1 to 500. Placing content in nodes can correspond for example to advertising educational content, link to online tutoring resources as well as online advertising.

The SCIENCE dataset: The SCIENCE dataset[5] contains of the citation network of articles that appeared in the domain of high-energy physics. The nodes in the network correspond to papers and the edges correspond to one paper citing the other. The purpose of this dataset is to depict the learning process of a person, who wants to obtain knowledge in high-energy physics. The person might read a paper and then based on the reference list of this paper choose his next read. Since this process does not necessary imply reading the papers in chronological order, we make edges bidirectional and from the resulting graph we pick the largest connected component. The end result of this preprocessing is a graph with 27K nodes and 704K edges.

The navigational models: Our navigational models are defined by the graph datasets described above. More specifically, using these graphs we define the navigational model for each dataset to be the corresponding *PageRank Markov chain* on the graph defined by the dataset. In these Markov chains, the user that is at node x chooses with probability α one of the outgoing links of x uniformly at random; with probability $(1-\alpha)$ the user jumps to a random node of the input graph. Note that our framework works for *any* Markov chain defined on the input navigational graphs, as long as the chain is ergodic. Our choice of the PageRank Markov chain [4] guarantees the ergodicity of our navigational models, even when the input graph is disconnected. For all our experiments we use $\alpha = 0.8$.

The conversion model: For all of our datasets we use a conversion model generated along the same principles. We first assign the initial CR values $C(0,s)$ to states in level 0. We obtain the CR rates $C(\ell, s)$ on subsequent levels with help of a function $f(\ell)$ based on formula (9)

$$C(\ell, s) = f(\ell)C(0, s). \tag{9}$$

The initial assignment of $C(0,s)$ to states is done by assigning to every state s value $C(0,s)$ chosen randomly among the following 10 candidate values:{0.2, 0.1, 0.07, 0.04, 0.03, 0.027, 0.018, 0.017, 0.015, 0.010}. Note that these values are exponentially decreasing and capture our intuition that there will be small number of states with high probability of affecting the conversion of the users. Experiments with different conversion values showed quite similar results; due to lack of space we do not report them.

For our experiments, we pick five different functions f. The types of functions depict our intuition of different possible user behaviors; one is linear increasing with ℓ, another is exponentially decreasing, and the last three are first increasing and then decreasing with different rates (linear, exponential, or a combination of the two). Every state s is assigned one of the above functions, which is then used for computing $C(\ell, s)$ for that state.

5.1.1 Baseline algorithms

Baseline algorithms: In order to better judge and understand the performance of our Greedy and Par-Greedy algorithms, we compare them with the following baselines.

Stationary: Given the navigational model $\mathcal{M}$ and budget B on how many states content can be place, the Stationary algorithm picks the B states with highest stationary distribution (highest PageRank).

Rank: The Rank algorithm is an extension of Stationary. That is, it first finds the stationary probability distribution $\pi(s)$ (i.e., the PageRank score) for every state s. Then, it computes the *rank* of each state s as $\mathit{rank}(s) = \pi(s) \times C(0,s)$. Given budget B the Rank algorithm picks the B nodes with the highest $\mathit{rank}(s)$ score.

Degree: For budget B, the Degree algorithm picks the B highest indegree states.

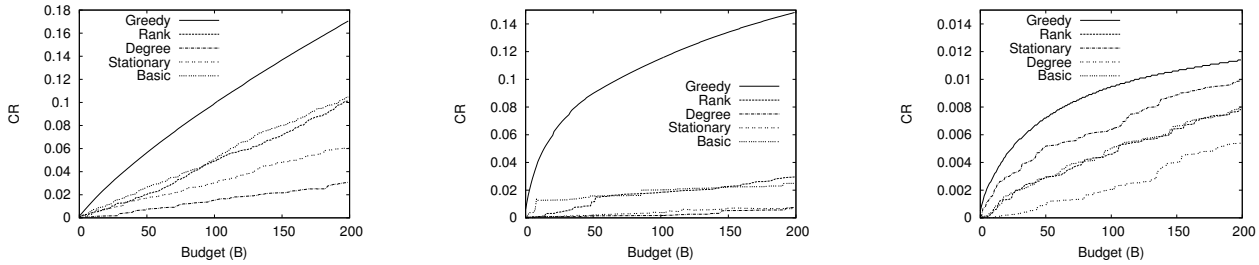
Basic: For budget B, we rank states $s \in \mathbf{S}$ in decreasing order of $C(\ell, 0)$, i.e., the probability of a user being converted at every state at level 0. Then, the Basic algorithm reports the top-B states from this order.

Note that comparison with Rand (i.e., the algorithm that

[3] www.cise.ufl.edu/research/sparse/matrices/Gleich/minnesota.html

[4] www.cise.ufl.edu/research/sparse/matrices/Gleich/wb-cs-stanford.html

[5] snap.stanford.edu/data/cit-HepTh.html



(a) ROAD network (b) WEB network (c) SCIENCE network

Figure 3: Real networks; x-axis: number of nodes where content is placed; y-axis: the expected `CR`.

selects random B nodes) is omitted since its performance is many orders of magnitude worse than any other algorithms.

5.2 Experimental results

In this section, we report the experimental results on the three real datasets we described above. Our results demonstrate the superior quality of the solutions obtained by `Greedy` (and `Par-Greedy`) and the scalability of our algorithms reasonably large datasets.

Qualitative evaluation: For the qualitative evaluation of our framework we run the `Greedy` algorithm as well as all the baseline algorithms (i.e., `Rank`, `Stationary`, `Degree` and `Basic`) for all our datasets and report the expected conversion rate of the solutions they obtained as a function of the budget $B = \{1, \ldots, 200\}$.

Results for the ROAD *and* WEB *datasets:* The results obtained for the ROAD and the WEB graph are shown in Figures 3(a) and 3(b). For both datasets we observe that `Greedy` gives clearly the best results when compared to all other baseline algorithms. Also in both datasets the ranking of the rest of the heuristics is consistent: `Basic` and `Rank` are the second best, with `Stationary` and `Degree` to follow with solution with much lower expected conversion rates. The fact that `Rank` gives better results than `Stationary` is expected since the former takes both the stationary distribution and the conversion probability of each node into account when forming its solutions, while the latter only looks at the stationary probability. Moreover, the relatively poor performance of `Degree` is also expected since the degree of the nodes is not necessarily correlated neither with its conversion probability nor its stationary distribution. Finally, the fact that `Basic` and `Rank` have almost identical performance is due to the fact that the $C(\ell, s)$ values are much larger (in scale) than the stationary probability values and, therefore, the choices of the two algorithms are very similar.

Although the general trends observed in both the ROAD and the WEB datasets are similar, one can also observe some high-level differences. More specifically, for the ROAD network (Figure 3(a)) the expected conversion rate of the solutions obtained by `Greedy` increases almost linearly with the size of the solution B. On the other hand, for the WEB dataset (Figure 3(b)) the increase appears to be steeper, i.e., the `Greedy` solutions appear to benefit tremendously by the addition of new nodes in the solution. We conjecture that this effect is a result of the "diversity" of the nodes in the WEB graph. That is, in the ROAD network there are no "special" nodes – after all nodes simply correspond to road intersections with an (average) degree equal to four. On the other hand, the nodes of the WEB dataset have larger diversity and the in-degrees of the nodes are distributed in a power-law fashion.

Results for the SCIENCE *dataset:* We also evaluate the performance of our methodology on even larger networks, we have experimented with SCIENCE dataset, which is an order of magnitude (in terms of the number of edges) larger than the other two. For this experiment, we have restricted our set of candidate nodes to a set of approximately 500 nodes sampled from the set of nodes in the graph with probability proportional to their in-degree. The expected conversion rate achieved by the solutions of the different algorithms are shown in Figure 3(c). For this experiment, we use the parallel implementation of `Greedy`, which we call `Par-Greedy`.

Although the superiority of our method is clear in this dataset as well, the relative performance of the other heuristics is different in SCIENCE; to see this compare the ranking of the baseline algorithms as indicated by the results in Figure 3(c) with the ranking obtained by the results for the ROAD and WEB datasets – shown in Figures 3(a) and 3(b). We can observe that for SCIENCE the performance of `Degree` is clearly the second best, giving solutions between than `Rank` and `Stationary`. Clearly part of this change is due to the fact that we are only considering a subset of the nodes as candidates for content placement. More over, this subset is selected in such a way that is biased towards nodes with high in-degree. As a result, among those candidates there are not so many nodes with high stationary probability *and* high conversion probability and, as a result, neither the `Rank` nor the `Stationary` algorithms perform as well as `Degree`. In fact, when we ran the experiment using all nodes as candidates, we would have obtained the same relative ranking for all baselines.

It is important to observe that in Figure 3(c) the expected conversion rate of the solutions obtained by our algorithm increases steeply as a function of the size of the solution. This steep increase resembles the results we obtained for the WEB dataset – shown in Figure 3(b). The explanation we had in that case, applies here as well: the nodes in the science graph (as well as the candidate nodes) have diverse degrees following a power-law like distribution.

Running times: In order to give an idea of the actual computational time required to run our experiments, we report here the running time required for executing one step of the `Greedy` algorithm, i.e., the execution of the for loop shown in line 3 of Algorithm 1.

Table 1 shows the running times of this execution both for the `Greedy` and `Par-Greedy` algorithms. Recall that `Par-`

`Greedy` executes the searching over the candidates in a parallel fashion. All the experiments were conducted using the configuration we described in the beginning of this section.

The results demonstrate that the parallel version of our algorithm offers an order-of-magnitude speedup to its corresponding serial implementation. Even further, the evaluation of our candidate set for the SCIENCE network was impossible to execute in the serial implementation of `Greedy`, but it was done in less than an hour using `Par-Greedy`.

Table 1: Running time of `Greedy` and `Par-Greedy` for the selection of the first node of their solution.

Dataset	`Greedy`	`Par-Greedy`
ROAD	131 secs	16 secs
WEB	7920 secs (2.2 hours)	670 secs
SCIENCE	n/a	2884 secs

6. CONCLUSIONS

In this paper, we introduced the RACP problem in navigational graphs, we studied its complexity and designed practical algorithms for solving it. The distinguishing feature of our work when compared to other existing work in information, transportation and navigational networks is that we consider memory-full user navigation, where the probability of a user adopting a content depends on the number of times she has seen the same content in the past.

The main technical contribution of the paper is the development of a framework for solving the RACP problem, while taking into account both the users' navigational and conversion models. The first model describes how the users navigate in the graph, while the other models the impact that repetition has on their inclination to adopt content placed on the graph nodes. The algorithmic framework we introduced here is flexible and can work for any navigational model that is described by a Markov chain and any conversion model that models the probability of a user adopting a content as a function of the state of the user and the number of her encounters with the content. Our algorithms exploit a connection between our model and random walks with absorbing states and provide a practical solution to the RACP problem. Our experimental evaluation on real datasets demonstrated the efficacy of these algorithms in multiple settings.

Acknowledgments: This research was supported in part by NSF grants CNS-1017529, IIS-1218437, CAREER-1253393, CISE/CNS-1239021, CISE/CNS-1012798 and ENG/EFRI-0735974 as well as gifts from Microsoft and Google.

7. REFERENCES

[1] D. Agarwal, B.-C. Chen, and P. Elango. Spatio-temporal models for estimating click-through rate. In *WWW*, pages 21–30, 2009.

[2] O. Berman, D. Krass, and C. Wei Xu. Generalized flow-interception facility location models with probabilistic customer flows. *Communications in Statistics. Stochastic Models*, 13(1):1–25, 1997.

[3] O. Berman, D. Krass, and W. Xu. Locating discretionary service facilities based on probabilistic customer flows. *Transportation Science*, 29, 1995.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, pages 107–117, 1998.

[5] R. F. Bruner. Repetition is the first principle of all learning. *Social Science Research Network*, 2001.

[6] M. Charikar, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. On targeting markov segments. In *STOC*, pages 99–108, 1999.

[7] N. Chen. On the approximability of influence in social networks. In *SODA*, pages 1029–1037, 2008.

[8] F. Chierichetti, R. Kumar, and P. Raghavan. Markov layout. In *FOCS*, pages 492–501, 2011.

[9] P. Domingos and M. Richardson. Mining the network value of customers. In *ACM SIGKDD*, pages 57–66, 2001.

[10] P. Doyle and J. Snell. *Random walks and electric networks*. Mathematical Association of America, 1984.

[11] E. Even-Dar and A. Shapira. A note on maximizing the spread of influence in social networks. In *WINE*, pages 281–286, 2007.

[12] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[13] M. J. Hodgson. A flow-capturing location-allocation model. *Geographical Analysis*, pages 270–290, 1990.

[14] M. J. Hodgson, K. Rosing, A. Leontien, and G. Storrier. Applying the flow-capturing location-allocation model to an authentic network: Edmonton, canada. *European Journal of Operational Research*, pages 427–443, 1996.

[15] J. Kemeny and J. Snell. *Finite Markov chains*. VanNostrand, New York, 1969.

[16] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence throw a social network. In *ACM SIGKDD*, pages 137–146, 2003.

[17] P. Kotler and G. Armstrong. *Principles of Marketing*. Pearson Education, Prentice Hall, 2005.

[18] J. Leskovec, A. Krause, C.Guestrin, C.Faloutsos, J.VanBriesen, and N.Glance. Cost-effective outbreak detection in networks. In *ACM SIGKDD*, pages 420–429, 2007.

[19] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *ACM SIGKDD*, pages 61–70, 2002.

[20] K. Tanaka and T. Furuta. Locating flow capturing facilities on a railway network with two levels of coverage. In *The Ninth International Symposium on Operations Research and Its Applications*, 2010.

[21] J. S. Trent, R. V. Friedenberg, and R. E. J. Denton. *Political Campaign Communication: Principles and Practices (Communication, Media, and Politics)*. 2011.

[22] C. J. Weibell. *Principles of learning: A conceptual framework for domain-specific theories of learning*. PhD thesis, Brigham Young University. Department of Instructional Psychology and Technology, 2011.