Local Algorithm for User Action Prediction Towards Display Ads

Hongxia Yang
Alibaba Group
Hangzhou, Zhejiang, China 311121
yang.yhx@alibaba-inc.com

Yada Zhu
IBM Research
Yorktown Heights, NY 10598
yzhu@us.ibm.com

Jingrui He
Arizona State University
Tempe, AZ 85281
jingrui.he@asu.edu

ABSTRACT

User behavior modeling is essential in computational advertisement, which builds users' profiles by tracking their online behaviors and then delivers the relevant ads according to each user's interests and needs. Accurate models will lead to higher targeting accuracy and thus improved advertising performance. Intuitively, similar users tend to have similar behaviors towards the displayed ads (e.g., impression, click, conversion). However, to the best of our knowledge, there is not much previous work that explicitly investigates such similarities of various types of user behaviors, and incorporates them into ad response targeting and prediction, largely due to the prohibitive scale of the problem.

To bridge this gap, in this paper, we use bipartite graphs to represent historical user behaviors, which consist of both user nodes and advertiser campaign nodes, as well as edges reflecting various types of user-campaign interactions in the past. Based on this representation, we study random-walk-based *local* algorithms for user behavior modeling and action prediction, whose computational complexity depends only on the size of the output cluster, rather than the entire graph. Our goal is to improve action prediction by leveraging historical user-user, campaign-campaign, and user-campaign interactions. In particular, we propose the bipartite graphs *AdvUserGraph* accompanied with the *ADNI* algorithm. *ADNI* extends the *NIBBLE* algorithm to *AdvUserGraph*, and it is able tofi nd the local cluster consisting of interested users towards a specific advertiser campaign. We also propose two extensions of *ADNI* with improved efficiencies.Th e performance of the proposed algorithms is demonstrated on both synthetic data and a world leading Demand Side Platform (DSP), showing that they are able to discriminate extremely rare events in terms of their action propensity.

CCS CONCEPTS

•Information systems →Clustering; •Theory of computation →Pattern matching; •Software and its engineering →Patterns;

KEYWORDS

User Action Prediction; Local Graph Algorithm; Large Scale; Computational Advertisement

ACM Reference format:
Hongxia Yang, Yada Zhu, and Jingrui He. 2017. Local Algorithm for User Action Prediction Towards Display Ads. In *Proceedings of KDD'17, August 13–17, 2017, Halifax, NS, Canada.*, , 9 pages.
DOI: http://dx.doi.org/10.1145/3097983.3098089

1 INTRODUCTION

Computational advertisement has been the subject of rigorous research with extremely fast development in the past decades.This area has generated billions of revenue, produced hundreds of scientific papers and patents, saw a broad variety of implementations, and yet the accuracy of state-of-the-art prediction technologies leaves to desire more. One essential component of computational advertisement is user behavior modeling, which builds users' profiles by tracking their online behaviors and then delivers the relevant ads according to each user's interests and needs. It is to be expected that similar users tend to have similar behaviors towards the displayed ads.Th erefore, by taking into consideration the similarity among different users, we may be able to improve the advertising performance. However, to the best of our knowledge, there is not much previous work that explicitly investigates such similarities and incorporates them into ad response targeting and prediction [2, 19].Th is is largely due to the prohibitive scale of the problem, usually exceeding billions of users.

To address this problem, in this paper, we propose to use bipartite graphs to represent historical user behaviors. Such graphs consist of two types of nodes: user nodes and advertiser campaign nodes.Th e existence of an edge between a user node and a campaign node indicates that the user interacted with the campaign related ads before either by clicking a page (a.k.a., click) or buying the advertiser's products (a.k.a., conversion) after seeing the ad (a.k.a., impression).Th erefore, similar users would have similar connections with the campaign nodes, and they are expected to have similar behaviors towards future ads. Based on such bipartite graphs, our task is to identify which users are likely to interact with a new ad in the future.Th e ultimate goal is to build predictive models to identify the potential customers for hundreds of different and concurrent display ad targeting campaigns. To this end, we propose to use local graph algorithms tofi nd the cluster consisting of interested users centered around the seed node corresponding to the advertiser campaign, as their computational complexity depends on the size of the cluster, which is significantly smaller than the entire graph. In particular, we propose the random-walk-based algorithm named *ADNI*, which is an adapted version of the *NIBBLE*

This work is supported by National Science Foundation under Grant No. IIS-1552654, ONR under Grant No. N00014-15-1-2821, and an IBM Faculty Award and part of the work is done when thefi rst author was working at Yahoo! Inc.Th e views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

algorithm [18] tailored for bipartite graphs. We also designed two extensions of *ADNI* for improved effectiveness and efficiency.

We evaluate the performance of *ADNI* and its extensions on a world leading advertising platform, *BrightRoll*, which is theflagship of *Yahoo!*'s programmatic ad buying application suite. It offers efficient Real Time Bidding (RTB) buying platforms and provides access to *Yahoo!* and third party inventories. *BrightRoll* capitalizes on relevant billions of user data each day and supports campaign management that capitalizes on maximizing ad campaign's reach, relevance and exposure frequency.Th us, it is a big challenge to set up afl exible and complete model framework that consistently integrates information from different dimensions, and our proposed algorithms are able tofi ll this gap.

The major contributions of this paper can be summarized as follows:

1. We propose *AdvUserGraph*, a novel bipartite graph representation for modeling user behaviors in computational advertisement.
2. We propose the *ADNI* algorithm tailored for *AdvUserGraph* with two extensions: *Approxi ADNI* and *Sparse ADNI* for identifying potential customers with respect to a specific advertisement campaign on such lopsided bipartite graphs (e.g., user nodes are much more compared to campaign nodes).
3. We use real-world data from a world leading advertising platform, *BrightRoll*, to empirically validate the effectiveness and efficiency of the proposed algorithms. In particular, we employ these algorithms in a user interest prediction task in cost per action (CPA) model for several campaigns with satisfactory performance.

The rest of the paper is organized as follows. In Section 2, we briefly review the related work on computational advertisement, conversion rate (CVR) prediction and graph-based modeling.Then in Section 3, we introduce *AdvUserGraph* accompanied with *ADNI* and its two sparse versions *Approxi ADNI* and *Sparse ADNI*. Section 4 presents experimental results on both synthetic data and *BrightRoll* applications. Finally, we conclude the paper in Section 5.

2 RELATED WORK

In this section, we briefly review the existing work on computational advertisement, CVR prediction and graph-based modeling.

2.1 Computational Advertisement

Computational advertisement is the foundation of building large scale automated systems to select ads in online advertising applications. An important goal is tofi nd the best match between a given bid request in a given context and a suitable ad. Different variations of the problem arise depending on the context considered. For example, one important objective of advertisers is to build brand awareness for promoting future sales, possibly targeting at a user segment.Th is is similar in spirit to advertising on television and magazines. Advertisers with this objective in mind usually opt for the Cost-per-Milli (CPM) model where impressions (user see ads on publisher pages) are priced in bundles of 1,000. In this scenario, "delivery" is the goal for publishers. If the advertisers' intentions are clicks and conversions(e.g., online course registration, credit card application or products purchase, etc), they usually adopt cost-per-click(CPC) or cost-per-action(CPA) models.Th e corresponding click-through-rate (CTR) and conversion-rate (CVR) are the two goals that we would like to optimize.

The CTR of an advertisement is defined as the number of clicks on an ad divided by the number of times the ad is shown (impressions), expressed as a percentage. Similarly, CVR is the proportion of conversions divided by the total number of impression (post view attribution only), or the total clicks (post click attribution only) or the total of both (post view and post click attributions). CVR prediction is even more challenging compared to CTR prediction since the data is much sparser.

2.2 CVR Prediction

Building conversion models is extremely challenging for many reasons and only a few papers have explored thisfi eld [4, 10, 16]. Usually, only a very small portion of the users that click or have been shown ads eventually convert and thus, conversions are very rare events.Th is constrains the modeling techniques to parsimoniously work with the data. On the other side, user profiles are high dimensional and sparse consisting of several different kinds of activities, ranging from user demographics to search queries and page browsing. Dealing with such different activities in the presence of limited conversion information is non-trivial. To add to this, the data is highly volatile due to cookie churn, changes in campaigns, variability in user interests and other temporal effects that do not allow accumulating long-standing data and require the modeling approach to have a quick start and dynamically adapt over time as new data accumulates. However CVR prediction is critical in realizing data driven targeted advertising [15, 21]. Essentially, CVR prediction is a probability regression problem where the positive instances are extremely sparse.Th us machine learning models with probability-related loss, such as logistic loss, are mainly used for user response prediction, including generalized linear regression [6, 9, 11], factorization machines [13], gradient boosting decision tree models [8] and the recent transfer learning [5, 14]. In this paper, we focus on tackling the CVR prediction which can be broadly divided to post view and post click conversions.

2.2.1 *Post View Conversion.*

View-through conversions correspond to people that converted after viewing an ad without having clicked the ad itself. It is often tracked via cookies that are placed on users' computers when the ad is served and via a conversion tracking code provided by the advertisers and placed on conversion pages. Most often, the view-trough conversion is counted when it happens within certain days after the user saw, but did not click, the ad. And, each conversion is credited only to the last ad impression being shown to the user.

2.2.2 *Post Click Conversion.*

Post click conversions correspond to people that converted after having clicked the ad.Th is is distinct from post view conversion since here the conversion has to occur subsequent to a click. But the conversion does not have to happen right after the user clicking through the ads. Similar to post view conversion, most often, the post click conversion is counted when it happens within certain days after the user clicked, and each conversion is credited only to the last click happening before the conversion.

2.3 Graph-Based Modeling

Today there exist innumerable applications which require the analyses of some type of large graphs, e.g., social networks, protein interaction networks, co-author networks, etc. Even more, the world wide web is estimated to contain at least 4.74 billion pages [1]. Consequently, the analyses of even moderately large networks are on the order of tens of thousands of vertices and pose significant challenges. One approach to deal with these problems is to partition such networks, cutting them into smaller, more manageable pieces which may be processed in parallel. However, the NP-complete problem offi nding an optimal clustering of vertices within one of these networks has been under investigation for over a decade [17].

Partitioning large graphs is indeed a computationally intensive problem: few methods exist which can partition a graph with n vertices and m edges in time that is close to even $O(n^2)$ or $O(m)$. A breakthrough in recent years has been the advent of local methods for graph partitioning, achieving a time complexity that is close to linear in the number of edges.Th efi rst of these methods was made possible by a local clustering algorithm called *NIBBLE* [18]. *NIBBLE* attempts to minimize the clustering quality metric *cut conductance* for undirected unweighted graphs. Given a starting vertex, it provablyfi nds a cluster near that vertex in time $(O(2^b \log^6 m)/\phi^4))$ that is proportional to the size of the output cluster. Finding a cluster in time proportional to its size is an extremely valuable routine in itself, and the authors show how *NIBBLE* can be used as a subroutine to repeatedly remove small clusters from a large graph in order to obtain a nearly-linear time graph partitioning algorithm.

Later [3] extended *NIBBLE* using PageRank vectors and showed that a sweep over a single PageRank vector canfi nd a cut with conductance ϕ, provided there exists a cut with conductance at most $\Omega(\phi^2/\log m)$ where m is the number of edges in the graph. By extending this result to approximate PageRank vectors, they develop an algorithm for local graph partitioning that can be used tofi nd a cut with conductance at most ϕ, whose small side has volume at least 2^b in time $O(2^b \log^3 m/\phi^2)$.

Convex optimization has become an increasingly popular way of modeling graphs in differentfi elds. However, as datasets get larger and more intricate, classical convex optimizations usually fail due to a lack of scalability. Recently [7] proposed the network Lasso and developed a fast, scalable and distributed solver and saw several successful applications in the graph related problems.

Besides the above mentioned directions, breakthroughs are also made from using physical models [20].Th is work presents a method that allows for the discovery of communities within graphs of arbitrary size in times that scale linearly with their size.Th is method avoids edge cutting and is based on notions of voltage drops across networks that are both intuitive and easy to solve regardless of the complexity of the graph involved. It is also showed how this algorithm allows for the swift discovery of the community surrounding a given node without having to extract all the communities out of a graph.Th e computational complexity of this algorithm is $O(m + n)$.

Both *NIBBLE* and PageRank *NIBBLE* are local algorithms, which find a solution containing or near a given vertex without looking at the whole graph.Th e running time of local algorithms, when finding a non-empty local cluster, is nearly linear in the size of the output cluster. [18] and [3] focus on unweighted and undirected massive graphs. [7] and [20] are global algorithms. In global clustering, each node of the graph is assigned a cluster in the output of the method. For large graphs such as social networks and web content graph, global approaches that require the entire graph to be accessible simultaneously do not scale well. In such settings, a more desirable approach is to use local clustering algorithms. However, neither *NIBBLE* nor PageRank *NIBBLE* directlyfi ts our needs where the **Size** of the recommended users is required beforehand.

3 THE PROPOSED FRAMEWORK

In this section, wefi rst introduce the bipartite graph(*AdvUserGraph*) representation of historical user behaviors together with the action importance defined on such graphs; then we present the proposed local algorithm(*ADNI*) forfi nding potentially interested users with respect to a specific advertiser campaign, followed by further sparse extensions(*Approxi ADNI* and *Sparse ADNI*).

3.1 Bipartite Graphs and Action Importance

In order to predict users' actions with respect to a specific campaign, in this paper, we propose to use bipartite graphs to represent historical user behaviors in terms of their interactions with advertiser campaigns. Such graphs are named *AdvUserGraph*. It contains two types of nodes, user nodes and campaign nodes.Th ere is an edge between a user node and a campaign node if and only if the user interacted with the campaign before via impression, click or conversion.Th erefore, given a campaign node, our goal is tofi nd a good cluster in the bipartite graph near this node with low conductance (introduced in the next subsection), such that the user nodes within this cluster are highly likely to interact with the campaign (if not in the past).

We consider both binary and continuous bipartite graphs, where edge weights can only be 0 or 1 in binary graphs and between 0 and 1 in continuous graphs. For binary graphs, we add an edge between two nodes (i.e., a user node and a campaign node) if any interaction (i.e., impression, click or conversion) exists in historical records. For continuous graphs, we simultaneously consider three types of edges, i.e., impressions, clicks and conversions.Th e raw weights of each edge will be the absolute number of impressions, clicks and conversions between users and campaigns.Th ere are usually a few tens of impressions but with a wide range, a few clicks with a smaller range and only one or two conversions if at all. However, the values of impressions, clicks and conversions are in the opposite direction, i.e., values(impression) $\ll$ values(clicks) $\ll$ values(conversion) ($\ll$ stands for much smaller). To deal with the above problem induced by the highly imbalanced distributions and values of the three edge types, we borrow the idea of tf-idf and make the following modifications. We assume the edges between each pair of user and campaign nodes to be a document and the three types of edges to be three unique terms. Denote $f(e_{ij}, d_i)$ the raw frequency of a term e_{ij} in a document d_i for $i = 1, \ldots, N$ where N is the total number of pairs of users and campaigns, and $j = 1, 2, 3$ the index of the three types of edges. We use the logarithmic scaled frequency to define the term frequency (tf) as follows:

$$\text{tf}(e_{ij}, d_i) = \begin{cases} 1 + \log f(e_{ij}, d_i), & f(e_{ij}, d_i) \neq 0 \\ 0, & \text{else.} \end{cases}$$

Then the inverse document frequency (idf) is defined accordingly as follows:

$$\text{idf}(e_{ij}, D) = \log \frac{N}{|\{d_i \in D : e_{ij} \in d_i\}|}$$

where N is the total number of documents in the corpus and $|\{d_i \in D : e_{ij} \in d_i\}|$ denotes the number of documents where term e_{ij} appears.Th efi nal tf-idf is calculated as

$$\text{tfidf}(e_{ij}, d_i) = \text{tf}(e_{ij}, d_i) \times \text{idf}(e_{ij}, D). \tag{1}$$

Later we use Equation (1) to define weights in the continuous *AdvUserGraph* andfi nd more promising results by discriminating different edge values compared to the binary *AdvUserGraph*.

3.2 ADNI Algorithm

In this subsection, we introduce the proposed *ADNI* algorithm for finding potentially interested customers with respect to a specific advertiser campaign. Formally, we study the clustering problem where the data set is given as an undirected graph represented by a similarity matrix: given an undirected graph $G = (V, E)$, we want tofi nd a set S that minimizes the relative number of edges going out of S with respect to the size of S (or the size of $\bar{S}$ if S is larger than $\bar{S}$, where $(S, \bar{S})$ is a partition of G). To capture this concept rigorously, [18] proposed the *cut conductance* of a set S as:

$$\phi_c(S) = \frac{|E(S, \bar{S})|}{\min\{\text{vol}(S), \text{vol}(\bar{S})\}}, \tag{2}$$

where $\text{vol}(S) = \sum_{v \in S} \text{d}(v)$, $\text{d}(v)$ represents the degree of the nodes and $E(S, \bar{S})$ is the set of edges connecting a vertex in S with a vertex in $\bar{S}$. So $\text{vol}(V) = 2|E|$. For binary graphs, degree of a node is defined as the number of edges connecting to the node; and for continuous graphs, it is defined as the sum of weights of connected edges. Finding S with the smallest $\phi_c(S)$ is called conductance minimization which is equivalent tofi nding the "best" cluster in the graph, if we rank clusters by their conductance.

The change in probability mass after one step of the random walk is a linear operator that is realized by multiplying the column vector of probabilities by the matrix

$$M = (BD^{-1} + I)/2, \tag{3}$$

where B is the adjacency matrix of the graph, D is the diagonal matrix with diagonal entries $(d(1), \ldots, d(n))$, with each diagonal element equals the row sum of B and I is the identity matrix. Following [18], we define

$$I(p, x) = \max_{\substack{w \in [0,1]^n \\ w(u)d(u) = x}} \sum_{u \in V} w(u)p(u), \tag{4}$$

where p is the distribution of the random walk over n entries. One can easily check that $I(p, 0) = 0$ and $I(p, 2m) = 1$. Let $S_j(p)$ be the set of j vertices u maximizing $p(u)/d(u)$ and denote $I_x(p, x)$ as the partial derivative of $I(p, x)$ with respect to x, we have

$$I_x(p, x) = \lim_{\delta \to 0} I_x(p, x - \delta) = \frac{p(\pi(j))}{d(\pi(j))}, \tag{5}$$

where $\pi(j) = S_j(p) - S_{j-1}(p)$ is the permutation function.Th us we have

$$\frac{p(\pi(i))}{d(\pi(i))} \geq \frac{p(\pi(i+1))}{d(\pi(i+1))}. \tag{6}$$

As $p(\pi(i))/d(\pi(i))$ is non-increasing, $I_x(p, x)$ is a non-increasing function in x and $I(p, x)$ is a concave function in x. $I(p, x)$ is used as one convergence measure and $I_x(p, x)$ characterizes the probability mass.

We generalize the *ADNI* in Algorithm 1. It is adapted from the *NIBBLE* algorithm proposed in [18], and is tailored for bipartite graphs. Constants c_1 to c_4 are defined in the same way as [18]. *ADNI* works as follows. It takes as input the bipartite graph G, the campaign seed nodes v_a, the lower bound k on the number of potentially interested customers, as well as the upper bound ϕ on the conductance of the local cluster, and outputs the set of k user nodes within the identified local cluster. In Steps 1 and 2, we initialize the parameters in the same way as [18]. Notice that in the bipartite graph G with n nodes, for an $n \times 1$ vector p and a positive constant ϵ, define $[p]_\epsilon$ to be an $n \times 1$ vector such that $[p]_\epsilon(v) = p(v)$ if and only if $p(v) \geq d(v)\epsilon$, where $d(v)$ is the degree of node v, and 0 otherwise. In other words, $[p]_\epsilon$ is a truncated version of p. Next, Steps 4 and 5 generate a sequence of vectors starting at r_0 by the following rule

$$q_t = \begin{cases} r_0, & \text{if } t = 0, \\ Mr_{t-1}, & \text{otherwise,} \end{cases}$$

where $r_t = [q_t]_\epsilon$, $t > 0$.Th at is, at each time stamp, we let the random walk proceed by one step from the current distribution and then round every $q_t(u)$ that is less than $d(u)\epsilon$ to 0. Notice that q_t and r_t are not necessarily probability vectors, as their components may sum to less than 1.Th en Step 7fi nds the set $S_j(q_t)$ consisting of j nodes whose corresponding elements in q_t are the largest, and Step 8 determines if this set contains the desired user nodes that correspond to potentially interested customers. In particular, it checks the following 3 conditions: condition **Size** in Step 9 guarantees that the output set has at least k recommended user nodes; condition **Volume** in Step 10 ensures that it contains a good amount of volume (e.g., not too much and not too little); condition **Large Prob Mass** in Step 11 guarantees that the output user nodes have a large probability mass. Notice that in condition **Large Prob Mass**, according to the definition of $I_x(p, x)$, $I_x(q_t, 2^b)$ can be computed as follows.

Remark3.1. *Assuming that j' satisfies $\lambda_{j'}(q_t) \leq 2^b \leq \lambda_{j'+1}(q_t)$, then*

$$I_x(q_t, 2^b) = \frac{q_t(\pi(j'))}{d(\pi(j'))}, \tag{7}$$

where we set $\lambda_j(p) = \sum_{u \in S_j(p)} d(u)$ and b to be the local cluster size.

Lemma3.2. *The time complexity for* ADNI *is bounded by* $O(2^b \frac{log^6 m}{\phi^4})$.

Proof. Similar to the *NIBBLE* algorithm, Algorithm 1 will run for up to t_{last} iterations. We will now show that each iteration takes time $O((k/\gamma - k + 1) \log m/\epsilon)$, where γ denotes the fraction

Algorithm 1 *ADNI* Algorithm

```
Input: $G, v_a, k, \phi, b$
Output: The set of $k$ user nodes within the local cluster
1: Compute $t_{last}$ and initialize $\epsilon$ according to [18] using $\phi$.
2: Initialize $r_0$ to be an $n \times 1$ all zero vector except for the element
   that corresponds to $v_a$, where $v_a$ denotes the seed nodes.
3: for $t = 1{:}t_{last}$ do
4:     Set $q_t = Mr_{t-1}$, where $M$ is defined as $\frac{1}{2}(BD^{-1} + I)$, $B$ is the
       $n \times n$ adjacency matrix of $G$, $D$ is the $n \times n$ diagonal matrix
       whose elements are set to be the degree of each node in $G$,
       and $I$ is the $n \times n$ identity matrix.
5:     Set $r_t = [q_t]_\epsilon$.
6:     for $j = k : n$ do
7:         Let $S_j(q_t)$ denote the set of $j$ nodes whose corresponding
           elements in $q_t$ are the largest.
8:         Return the $k$ user nodes in $S_j(q_t)$ as the ranked list, and
           quit if the following conditions are satisfied
9:          – Size: the number of user nodes in $S_j(q_t)$ is at least $k$.
10:         – Volume: $2^b \le \lambda_j(q_t) < \frac{5}{6}\mathrm{vol}(G)$.
11:         – Large Prob Mass $I_x(q_t, 2^b) \ge \frac{1}{c_4}(l+2)2^b$.
12:     end for
13: end for
14: Return an empty set.
```

of user nodes among all the nodes in the graph. l, t_{last} and ϵ are defined in the same way as [18]:

$$l = \lceil \log_2(\mu(V)/2) \rceil \tag{8}$$

$$t_{last} = (l+1)\left\lceil \frac{2}{\phi^2}\log(c_1(l+2)\sqrt{\mu(V)/2})\right\rceil \tag{9}$$

$$\epsilon = 1/(c_3(l+2)t_{last}2^b) \tag{10}$$

Let V_t represent the set of vertices such that $\forall u \in V_t$, $r_t(u) > 0$, which can be computed in time $O(|V_t|)$ in Step 5. Given the knowledge of V_{t-1}, the multiplication in Step 4 can be performed in time proportional to

$$\mu(V_{t-1}) = \sum_{u \in V_{t-1}} d(u) \le \sum_{u \in V_{t-1}} r_t(u)/\epsilon \le 1/\epsilon. \tag{11}$$

Steps 7 to 11 require sorting the vectors in V_t according to r_t, which takes time at most $O(|V_t|\log n)$.Th us, the running time for the inner loop of Algorithm 1 is bounded by $O((k/\gamma - k + 1)\log m/\epsilon)$. In our application, $\gamma \approx 1$.Th us, putting everything together, the running time of *ADNI* is bounded by

$$O\left(t_{last}(k/\gamma - k + 1)\log n/\epsilon)\right) \approx O\left(2^b \frac{\log^6 m}{\phi^4}\right). \tag{12}$$

□

There are two major differences between the proposed *ADNI* algorithm and the *NIBBLE* algorithm in [18]. First, in *ADNI*, the identified local cluster $S_j(q_t)$ does not necessarily have a conductance lower than ϕ, and yet it is thefi rst identified local cluster that satisfies all 3 conditions (Size, Volume, Large Prob Mass) via truncated diffusions. When applying the *ADNI* algorithm, ϕ is typically chosen based on the known advertiser campaign/user interactions similar to the target advertiser campaign v_a, and it is only used to compute t_{last} and ϵ according to [18]. Second, in *ADNI*, instead of using the local cluster size b as the input, we use the number of user nodes k in the local cluster. In this way, we allow other advertiser nodes to be included in the local cluster, which may provide us with more insights regarding the similarity of advertiser campaigns with respect to users' interactions.

3.3 Two Further Extensions

The following two steps are most time consuming when applying *ADNI* on large graphs: (1) ranking q_t to obtain the largest j nodes in $S_j(q_t)$ and $I_x(q_t, 2^b)$; (2) the inner loop between Steps 6 and 12 of *ADNI* if $k \ll n$. To further improve the running time of *ADNI*, we revise the inner loop of *ADNI* and propose *Approxi ADNI*. In *ADNI*, $S_j(q_t)$ is obtained by fully sorting the n nodes with time complexity at least $O(n\log(n))$. Instead in *Approxi ADNI*, we use the partition algorithm *introselect* [12] to determine $S_j(q_t)$ in Step 7 and select j nodes whose corresponding nonzero elements in q_t are largest without sorting all the elements.Th is approximation greatly reduces the time complexity to $O(|V_t|)$, where V_t is the set of nodes at which $r_t(u) > 0$ and $O(|V_t|) \le O(n)$.

On the other hand, due to the cold-start property of the motivating application, the input graph $G = (V, E)$ can be very sparse. To further improve the computational efficiency, we modify the implementation of *Approxi ADNI* to a sparse version, named *Sparse ADNI*. In particular, we define D as a sparse $n \times n$ diagonal matrix and I as the sparse $n \times n$ identity matrix. As a result M becomes a truncated sparse matrix and r_t is a sparse vector. All the computation in the inner loop only involves non-zero elements in q_t and r_t.

4 EXPERIMENTAL RESULTS

In this section, we demonstrate the performances of the proposed *ADNI* algorithm and its extensions with both binary and continuous bipartite graphs *AdvUserGraph*. In particular, we aim to answer the following two questions:

(1) What are the benefits that we bring in for *ADNI*, *Approxi ADNI* and *Sparse ADNI* from adapting the original *NIBBLE* on the applications of user action prediction through studying *AdvUserGraph*? We answer this question in Section 4.1.
(2) How do the performances of *ADNI* and its extensions compare with state-of-the-art techniques, including *NIBBLE* [18], PageRank *NIBBLE* [3], network Lasso [7] and Voltage [20] on real massive graphs? We provide detailed analyses in Section 4.2.

4.1 Synthetic Data

4.1.1 Data Generation. We generate a synthetic bipartite graph of 10M user nodes and 1K campaign nodes that share similar characteristics as real data. For each user and campaign pair, the generation of impressions, clicks, and conversions are consistent with real data: # impressions ≫ # clicks ≫ #conversions; the campaign CTR is around 1% and CVR around 0.1%. For the synthetic binary bipartite graph, the existence of an edge depends on whether there is any interaction (a.k.a., impression, click or conversion) between the

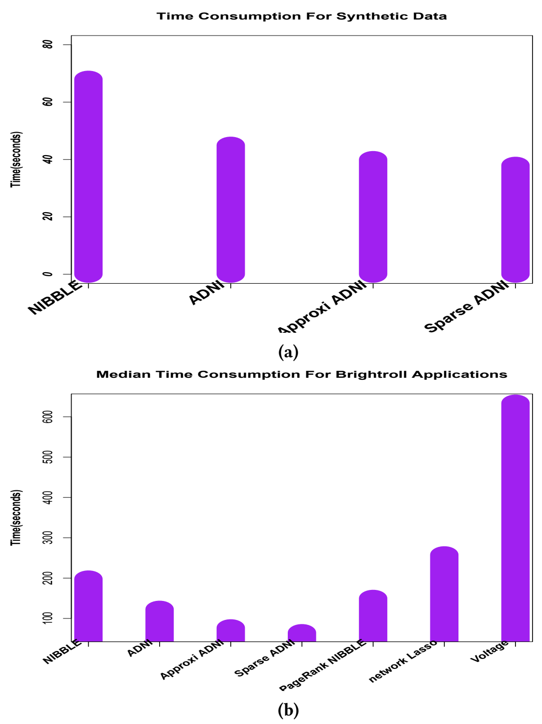


Figure 1: Efficiency Comparison on Both Synthetic and Data Collected from *BrightRoll* Data

user and campaign nodes. For the synthetic continuous bipartite graph, edge weight is calculated using Equation (1).

To test both the effectiveness and efficiency of *ADNI*, *Approxi ADNI* and *Sparse ADNI*, we use a three by three factorial design. One experimental factor is if replacing the cut conductance condition in the original *NIBBLE* by the **Size** requirement as in *ADNI* will boost the algorithm efficiency while still achieving similar effectiveness in massive bipartite graphs.Th e second design factor represents "sorting exclusive" variant. For the "sorting exclusive" variant, we adopt *ADNI* (Algorithm 1) and for the "no sorting exclusive" variant, we use *Approxi ADNI* (Algorithm 2).Th e third design factor is the sparse estimation of M. For the "sparse" variant, we use the sparse version of *ADNI* or *Sparse ADNI* (Algorithm 3) and for the "non-sparse" variant, we will again use *ADNI*. Both *Approxi ADNI* and *Sparse ADNI* are approximated versions of *ADNI* on some aspects and we would like to see if they can lead to improved efficiency with little side effect of effectiveness.

4.1.2 Evaluation Metrics. In order to comprehensively investigate both the efficiency and effectiveness of *ADNI* and its extensions, we focus on 3 measures. We use running time to compare the efficiency, cut conductance and precision/recall to characterize the effectiveness. We denote each learnt subgraph as S and the simulated underlying ground truth as A, and define precision and recall as follows:

$$\text{precision} = \frac{|A \bigcap S|}{|S|}, \quad \text{recall} = \frac{|A \bigcap S|}{|A|}.$$

4.1.3 Result Analyses. We learn the subgraph of each campaign node with the size requirement (500K recommended user nodes) and each of the following reported measure is based on the median values over the 1K campaign nodes. Time complexities are theoretically shown in Lemma 3.2. Here we report the running time (Figure 1a) for the synthetic bipartite graph which has around 10M nodes and 1M edges. On average *ADNI* consumes around 67% of the time needed by *NIBBLE*. Furthermore, *Approxi ADNI* and *Sparse ADNI* reduce the running time by approximately 10% compared with *ADNI*; and *Sparse ADNI* is also more efficient compared to *Approxi ADNI*. Notice that it is very hard for *NIBBLE* to control the size of the output sets and we have to loop and relax over ϕ to achieve the size requirement.Th is is detrimental to its efficiency and even more severe in practice. In conclusion, we observed significant efficiency improvement of *Approxi ADNI* and *Sparse ADNI* over *NIBBLE*.

Next we report pairwise comparisons of the cut conductance in Figure 3a for the binary bipartite graph and Figure 3b for the continuous bipartite graph. To better evaluate the incremental improvements induced by *ADNI*, *Approxi ADNI* and *Sparse ADNI*, we report pairwise comprisons here. For each campaign node, let A denote the ground truth graph, S the subgraph learnt from *NIBBLE*, S^0 from *ADNI*, S^1 from *Approxi ADNI* and S^2 from *Sparse ADNI* respectively. Notice that it is possible to see an output set S to have smaller cut conductance than A, because A is not necessarily the sparest cut in the graph. Recall that smaller cut conductance represents a sparer cut in the graph or better performance. Compared to *NIBBLE*, *ADNI* achieves sparser cuts (or $\phi(S^0)/\phi(S) < 1$) for around 50% campaign nodes in the binary synthetic graph and 80% in the continuous synthetic graph. *ADNI* achieves around 10% more sparser cuts compared to *Approxi ADNI* (or $\phi(S^0)/\phi(S^1) < 1$) and 15% more sparser cuts compared to *Sparse ADNI* (or $\phi(S^0)/\phi(S^2) < 1$). In general, $\phi(S^2) < \phi(S^1) < \phi(S^0) < \phi(S)$. We also observe the way that we define the edge weights in the continuous graph brings us even more improvements, where *ADNI* and its extensions show more advancements in continuous bipartite graphs compared to binary bipartite graphs (in Figure 3b). To conclude, even we do not explicitly require the constraint of ϕ in *ADNI* and its two extensions, the outputs still achieve similar ϕ in the binary bipartite graph and better ϕ in the continuous bipartite graph.

We report precision/recall in Figure 4a for the binary graph and Figure 4b for the continuous graph. In our scenario precision is the conversion rate of the recommended users while recall is the fraction of the converters that are included in the recommended users out of the total converters. Precision/Recall range between 0 and 1 and the larger the better. Again, *ADNI*, *Approxi ADNI* and *Sparse ADNI* all achieve very promising results.Th ere is almost no performance difference between *ADNI* and *NIBBLE*. *ADNI* performs better compared to *Approxi ADNI* and *Sparse ADNI* on average. Similar as before we see more improvement by using the continuous graph compared to the binary graph.

In conclusion, *ADNI*, *Approxi ADNI* and *Sparse ADNI* algorithms greatly outperform *NIBBLE* regarding efficiency which is critical in real time deployment. Besides, they achieve similar effectiveness regarding cut conductance and precision/recall for both binary and continuous graphs. All three exhibit even more improvements in the continuous graph than the binary graph. If we think about these methods as global strategies for all campaigns, we can conclude with strong confidence that *ADNI* and its extensions achieve the same or even better results in massive graphs compared to *NIBBLE* though we replace the cut conductance condition in our scenario to guarantee the practical **Size** requirement.

4.2 Brightroll Applications

4.2.1 Data Collection. In the second experiment, we test *AdvUserGraph* on the real motivating problem of action prediction in computational advertisement and show benefits induced by the bipartite graph in different campaign categories. We expect that some campaign categories will benefit more if they have more user stickiness. Wefi rst collect 7 days of the users' interaction history from 50 campaigns that are currently served by *BrightRoll* and there are in total close to 1B impressions, 2.5M clicks and 1M conversions. These campaigns belong to 10 different categories, including Entertainment, Automotive, Business, Education, Parenting, Health, Food/Drink, Home/Garden, Law/Politics and News. For *ADNI* and its extensions, we report results from the continuous bipartite graph since in reality they show much better performance compared to the binary version, as accounting for the importance of different kinds of edges is critical.

4.2.2 Evaluation Metrics. We test *ADNI*, *Approxi ADNI* and *Sparse ADNI* together with *NIBBLE*, PageRank *NIBBLE*, network Lasso and Voltage to recommend the top 10M users for each campaign. To briefly recap (refer to Section 2.3 for more details), *NIBBLE* [18] attempts to optimize cut conductance for undirected unweighted graphs and it provablyfi nds a cluster near the starting vertex in time that is proportional to the size of the output cluster. [3] extends *NIBBLE* and improves the efficiency of the original *NIBBLE* using PageRank vectors. [7] proposed the network Lasso and developed a fast, scalable and distributed solver that saw several successful applications on large graphs. [20] proposed the Voltage model which avoids edge cutting and is based on notions of voltage drops across networks. It also allows for the discovery of communities within graphs of arbitrary size with running time that scales linearly with respect to their size. Both *NIBBLE* and PageRank *NIBBLE* are local algorithms and the network Lasso and Voltage model are global algorithms. In order to compare efficiency and effectiveness, we focus on the following measures that are most critical for *online* performances:

1. Running time.
2. AUC: the algorithms' area under the ROC (Receiver Operating Characteristic) curve, which is usually used to quantify the quality of the predicted ranking that results from the algorithm according to the predicted probability.
3. CVR: we recommend 10M users for each campaign and calculate the proportion that will be converted in the next 3 days.The recommended users can be overlapping and the conversions will be assigned eqaully to those algorithms recommend them initially. 10M is approximated through the daily budgets, eCPA (expected Cost Per Action) requirement and audience sizes. It can vary from campaign to campaign, but 10M is a reasonable number for most of the campaigns.
4. BPI: total spending and eCPA are the two most important criteria to characterize revenue. In order to quantify the performance that can reflect these two measures in a consistent way, we propose the following Business Performance Index (BPI):

$$\text{BPI} = \frac{rev.test + (cost.ctrl - cost.test)}{rev.ctrl}. \tag{13}$$

where $rev.test$ and $rev.ctrl$ are calculated through number of conversions multiplied by CPA goal and $cost.test$ and $cost.ctrl$ are mainly inventory costs.

4.2.3 Result Analyses. First we report the median running time of the 50 campaigns in Figure 1b from the 7 competitors. Notice that *NIBBLE*, PageRank *NIBBLE*, *ADNI* and its two extensions are local algorithms. Voltage and network Lasso are global algorithms that take into consideration the entire graph. Overall, local algorithms are much more efficient compared to global algorithms on the massive bipartite graphs of computational advertisement. Among the local algorithms, *Sparse ADNI* achieves the most efficiency. Among the global algorithms, network Lasso almost triples the running time of *Approxi ADNI* and *Sparse ADNI*. Among the 7 algorithms, Voltage is the least efficient.

Besides the efficiency, we also report AUC, CVR and BPI to characterize the effectiveness of different algorithms.Th ese results are reported in Table 1. We use bold blue colors to highlight wins and bold red colors losses. *ADNI* beats the other 6 regarding all three measures, and its two extensions *Approxi ADNI* and *Sparse ADNI* rank next. Global algorithms that require the entire graph to be accessible simultaneously do not perform well, since it is extremely hard for global algorithms to control the size of output sets, which is detrimental for the performance. Voltage again achieves the worst results and this is probably because voltage drops do notfit well for the user-campaign linkage study. In conclusion, local algorithms not only achieve the best efficiency but also the promising effectiveness as well.

To further test the effectiveness induced by the bipartite graph *AdvUserGraph* for different campaign categories, we also study the performance by categories and report the category level results in Figure 2 in the form of a pie chart. We define the length of the pie associated with the category with the best AUC value (Food/Drink) as 1 and the length of other pies are obtained proportionally.The width represents the sample proportions of the 50 campaigns.Th ere are two observations here: 1. Different campaign categories indeed show different improvements by including the *AdvUserGraph*. And we believe that the more user stickiness (e.g., Food/Drink, Auto, Entertain), the more improvements *AdvUserGraph* can bring; 2.The quality of the *AdvUserGraph* is improved as more data becomes available.

5 CONCLUSIONS

Motivated by applications of large-scale action prediction for billions of users in computational advertising, we propose *AdvUserGraph*, a novel bipartite graph representation to tackle the CVR prediction problem in computational advertisement for user ranking and recommendation to campaigns. In particular, we propose an *ADNI* algorithm with two further extensions *Approxi ADNI* and *Sparse ADNI*, where are random-walk based *local* algorithm whose running time depends on the size of the output cluster. Unlike traditional approaches to clustering, which attempt to minimize a global metric designed to compare different clustering of the entire graph, the *ADNI* algorithmfi nds a cluster near an input node

Candidates	AUC	CVR	BPI
NIBBLE	0.67 (0.11)	9.89% (2.34%)	1.32 (0.34)
ADNI	**0.72(0.07)**	**12.10% (1.92%)**	**1.56 (0.28)**
Approxi ADNI	0.69 (0.10)	10.92% (3.01%)	1.42 (0.35)
Sparse ADNI	0.68 (0.12)	10.11% (3.78%)	1.38 (0.43)
PageRank *NIBBLE*	0.64 (0.10)	9.93% (4.02%)	1.29(0.44)
Voltage	**0.53 (0.13)**	**5.14 % (2.37%)**	**0.88 (0.33)**
network Lasso	0.64 (0.20)	8.32% (3.92%)	1.20 (0.46)

Table 1: Comparison results for seven competitors. We report median values from the 50 campaigns and relative variances in the parentheses. We use blue colors to highlight wins and red colors for loss.

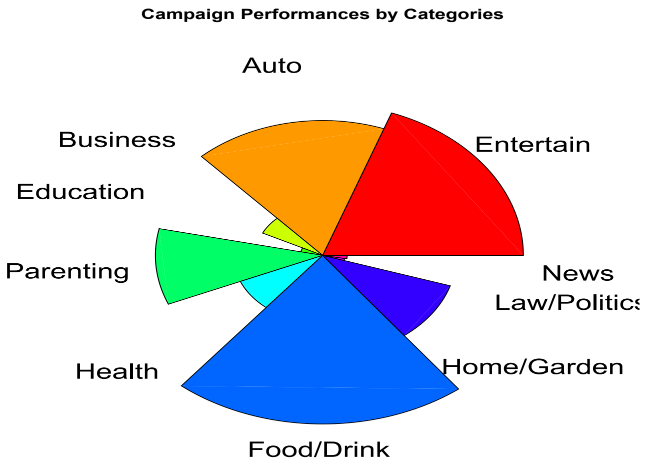
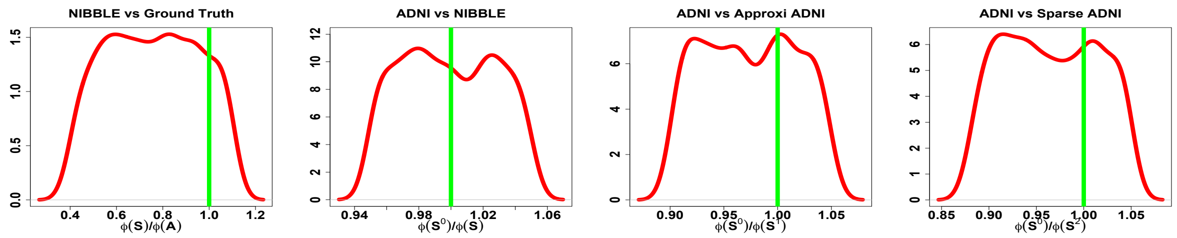


Figure 2: Campaign Performances by Category.Th e length of the best category Food/Drink is 1 and other categories are proportional. Widths of the category slices represent the sample proportion of each category.

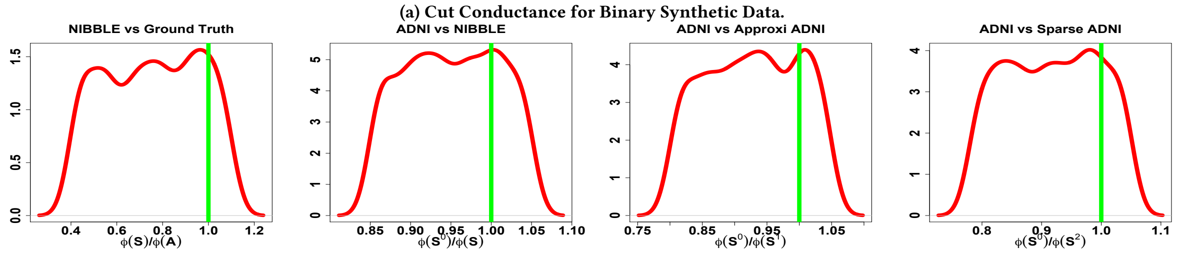
by only looking at a small neighborhood of that node within the graph. It is rather remarkable that even with a very limited view of the graph, we are able tofi nd a good cluster of vertices, which is extremely beneficial for the huge graph that we are dealing with in computational advertisement. Experimental results on both synthetic data and real data from *BrightRoll* show that the proposed easily-implemented, random-walk based local algorithms strike a good balance between effectiveness and efficiency, thus is more appropriate for today's data-intensive applications of computational advertisement compared with state-of-the-art techniques.

REFERENCES

[1] The size of the world wide web. https://www.worldwidewebsize.com.
[2] F. Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 273–280, New York, NY, USA, 2013. ACM.
[3] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 475–486, 2006.
[4] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *ACM Trans. Intell. Syst. Technol.*, 5(4):61:1–61:34, 2015.
[5] B. Dalessandro, D. Chen, T. Raeder, C. Perlich, M. Han Williams, and F. Provost. Scalable hands-free transfer learning for online advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1573–1582, 2014.
[6] T. Graepel, J. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on Machine Learning*, ICML '27, 2010.
[7] D. Hallac, J. Leskovec, and S. Boyd. Network lasso: Clustering and optimization in large graphs. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 387–396, 2015.
[8] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Candela. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ADKDD'14, pages 5:1–5:9, New York, NY, USA, 2014. ACM.
[9] K. Lee, B. Orten, A. Dasdan, and W. Li. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 768–776, New York, NY, USA, 2012. ACM.
[10] M. Mahdian and K. Tomak. Pay-per-action model for online advertising. *Internet and Network Economics, Lecture Notes in Computer Science*, 4858:549–557, 2007.
[11] H. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. Hrafnkelsson, T. Boulos, and J. Kubica. Ad click prediction: A view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1222–1230, New York, NY, USA, 2013. ACM.
[12] D. R. Musse. Introspective sorting and selection algorithm. *Software: Practice and Experience (Wiley)*, 27(8):983–993, 1997.
[13] R. Oentaryo, E. Lim, J. Low, D. Lo, and M. Finegold. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 123–132, New York, NY, USA, 2014. ACM.
[14] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost. Machine learning for targeted display advertising: transfer learning in action. *Machine Learning*, 95:103–127, 2014.
[15] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 521–530, New York, NY, USA, 2007. ACM.
[16] R. Rosales, H. Cheng, and E. Manavoglu. Post-click conversion modeling and analysis for non-guaranteed delivery display advertising. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 293–302, 2012.
[17] S. Schaeffer. Survey: Graph clustering. *Journal Computer Science Review*, 1(1):27–64, 2007.
[18] D. Spielman and S. Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal of Computation*, 42:1–26, 2013.
[19] K. Verstrepen and B. Goethals. Unifying nearest neighbors collaborativefiltering. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 177–184, New York, NY, USA, 2014. ACM.
[20] F. Wu and B. A. Huberman. Finding communities in linear time: A physics approach. *European Physical Journal B*, 38:331–338, 2004.
[21] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, New York, NY, USA, 2009. ACM.

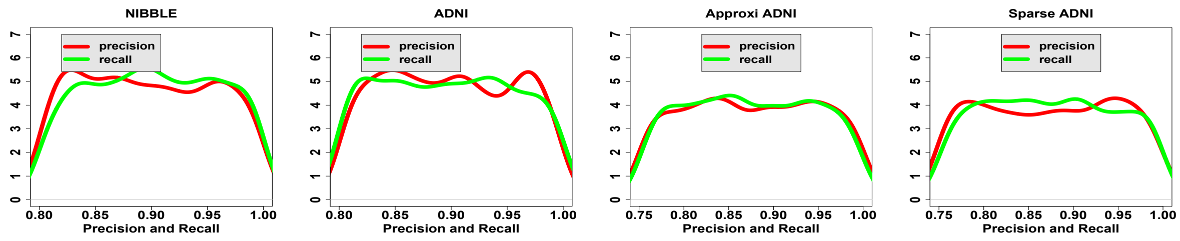


(a) Cut Conductance for Binary Synthetic Data.

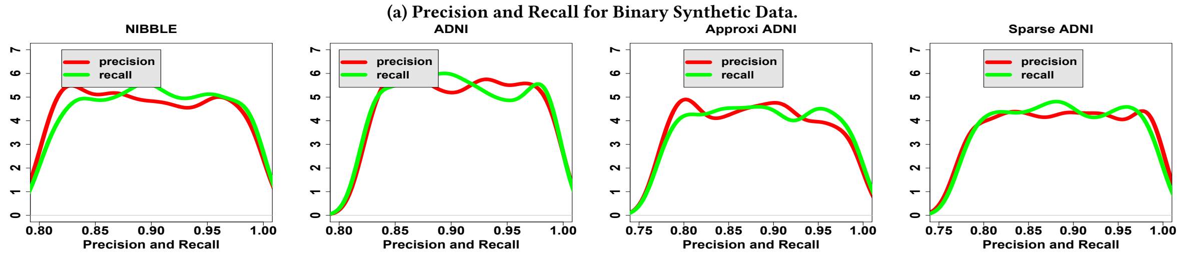


(b) Cut Conductance for Mixed Synthetic Data.

Figure 3: Cut Conductance Comparison on the Synthetic Data. A denotes the ground truth graph, S the subgraph learnt from *NIBBLE*, S^0 from *ADNI*, S^1 from *Approxi ADNI* and S^2 from *Sparse ADNI* respectively. ϕ stands for cut conductance and ratios that are smaller than 1 indicates sparser cuts. To conclude, in binary bipartite graph, *NIBBLE* and *ADNI* achieve relatively the same sparse cuts, both outperform *Approxi ADNI* and *Sparse ADNI*. In continuous bipartite graph, *ADNI* performs the best and the other three perform relatively the same.



(a) Precision and Recall for Binary Synthetic Data.



(b) Precision and Recall for Mixed Synthetic Data.

Figure 4: Precision/Recall Comparison on the Synthetic Data. In our scenario precision is the conversion rate of the recommended users while recall is the fraction of the converters that are included in the recommended users out of the total converters. Precision/Recall range between 0 and 1 and the larger the better. *ADNI* and *NIBBLE* perform relatively the same and both perform better compared to *Approxi ADNI* and *Sparse ADNI* on average. We also see more improvement by using continuous graph compared to binary graph.