A Graph-based Push Service Platform

Huifeng Guo[1]*, Ruiming Tang[2], Yunming Ye[1]**, Zhenguo Li[2], and Xiuqiang He[2]

[1] Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China
huifengguo@yeah.net, yeyunming@hit.edu.cn
[2] Noah's Ark Lab, Huawei, China
{tangruiming, li.zhenguo, hexiuqiang}@huawei.com

Abstract. Learning users' preference and making recommendations is critical in information-exploded environment. There are two typical modes for recommendation, known as *pull* and *push*, which respectively account for recommendation inside and outside the item market. While previously most recommender systems adopt only pull-mode, push-mode becomes popular in today's mobile environment. This paper presents a push recommendation platform successfully deployed for Huawei App Store, which has reached 0.3 billion registered users and 1.2 million Apps by 2016. Among the various modules in developing this push platform, we recognized the task of target user group discovery to be most essential in terms of CTR. We explored various algorithmic choices for mining target user group, and highlighted one based on recent advance in graph mining, the Partially Absorbing Random Walk [13], which leads to substantial improvement for our push recommendation, compared to the state-of-the-art including the popular PageRank. We also covered our practice in deploying our push platform in both single server and distributed cluster.

Keywords: Partially Absorbing Random Walk, Push Recommendation

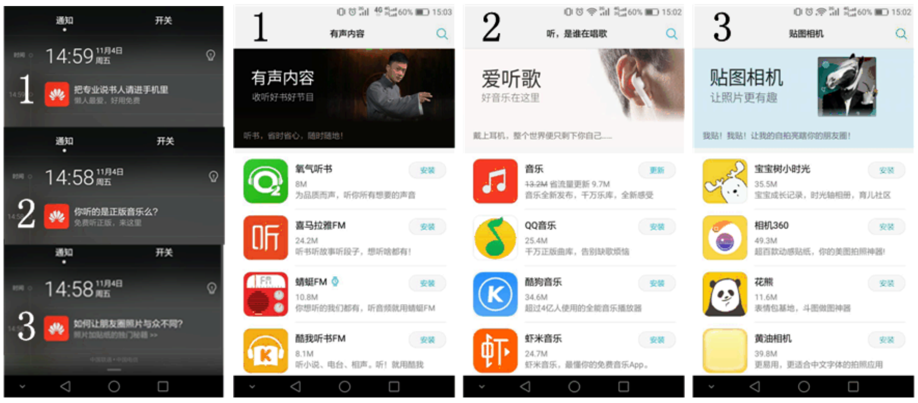
1 Introduction

With the rapid development of the Internet and mobile devices, our daily life connects closely to online services, such as online shopping, online news and videos, online social networks, and many more. In such highly dynamic, information-exploded environment, it is crucial to learn the preference of users and make recommendations accordingly.

Recommendation often comes in one of the two modes, the *pull-mode* and *push-mode*. The pull-mode recommends items to users *after* users enter the item market. The push-mode pushes items to users proactively *before* the users enter the item market. Compared to pull recommendation, push recommendation can offer two unique advantages: to rebuild connection with users for the service provider and to enhance experience for the users – a user can be informed of relevant items anytime, without entering the item market. Unlike pull-mode that

* The work is done when Huifeng Guo works as an intern in Noah's Ark Lab, Huawei.
** Corresponding author.

selects items for all users who are visiting the item market, the key in push recommendation is to identify a relatively small set of potential users for a given set of items. Unfortunately, techniques developed for pull recommendation such as matrix factorization are no longer suitable for the push-mode scenario, due to the following "cold-start" challenges: 1) the items to be pushed are usually new, with limited information available; and 2) "semi-active" and "inactive" users[3] rarely interact with the item market, and therefore not much of their information is available. While pull recommendation has been studied extensively, push recommendation is a new research area, especially to the academic community. In this paper, we present a Push Service Platform for Huawei App Store, one of the most large-scale and influential App markets in the world.



把专业说书人请进手机里
懒人最爱，好用免费

你听的是正版音乐么?
免费听正版，来这里

如何让朋友圈照片与众不同?
照片加贴纸的独门秘籍 >>

有声内容
收听好书好节目
听书，省时省心，随时随地！

氧气听书 8M 为品质而声，听你所有想要的声音
喜马拉雅FM 24.2M 听书听故事听段子，想听啥都有！
蜻蜓FM 10.8M 你想听的我们都有，听音频就用蜻蜓FM
酷我听书FM 8.1M 听小说、电台、相声。听！就用酷我

爱听歌
好音乐在这里
戴上耳机，整个世界便只剩下你自己....

音乐 ~~13.2M~~ 省流量更新 9.7M 音乐全新发布，千万乐库，全新感受
QQ音乐 25.4M 千万正版曲库，告别缺歌烦恼
酷狗音乐 34.6M 超过4亿人使用的全能音乐播放器
虾米音乐 24.7M 虾米音乐，最懂你的免费音乐App。

贴图相机
让照片更有趣
我贴！我贴！让我的自拍亮瞎你的朋友圈！

宝宝树小时光 35.5M 宝宝成长记录，时光轴相册，育儿社区
相机360 49.3M 超百款动感贴纸，你的美图拍照神器!
花熊 11.6M 表情包基地，斗图做图神器
黄油相机 39.8M 更易用，更适合中文字体的拍照应用

(a) Push message (b) Book listening (c) Music (d) Photo editor

Fig. 1: Push Services for Huawei App Store

Figure 1 shows three push activities in Huawei App Store, book listening, music, and photo editor. Through the messages from the notification center (Figure 1a), semi-active or inactive users are well informed of potentially relevant Apps without entering Huawei App Store. They can download their favorite Apps in the display pages by just clicking on the push message (Figures 1b, 1c, 1d). Behind such convenience in connecting services to users, what is the key enabling technology?

During our extensive practice in establishing the push service for Huawei App Store, we found that identifying the right users to target is the most challenging task because too many unrelated messages could disturb users and degrade the experience. Another challenge comes from the large scale of the problem. With the versatility of smart phone and various needs from our daily life, a large number of Apps are being created by developers and installed by users. In Huawei App Store, there are 0.3 billion registered users and 1.2 million Apps by 2016. For a service (App) to be push, how to identify the users of interest from such a web-scale user pool? Especially, on average based on our statistics, for each

[3] In Application Market, "active users" refers to the users who visit frequently, "inactive users" are those who do not visit recently, and "semi-active users" are those who do not visit often recently.

service, there are less than 1% of the population relevant to the service, making the target user discovery extremely difficult.

In response to these practical challenges, we have established a Push Service Platform (*PSP*) for Huawei App Store, which mainly consists of three layers: distributed storage layer, application layer, and evaluation layer. The contributions of this paper are summarized as follows:

- We present a Push Service Platform for Huawei App Store. Particularly, we identify the target user discovery problem as the most significant task for the push service.
- We carefully compare different choices of algorithms for mining target user group, and highlight in details one based on recent advance in graph mining, namely Partially Absorbing Random Walk [13], which has been adopted by our push service. Particularly, we propose and implement an approximate partially absorbing random walk algorithm (A-PARW) for both single server and distributed cluster that can support very large-scale problems and can efficiently respond to a multitude of push services simultaneously.
- We conduct off-line and on-line experiments in Huawei App Store which shows that A-PARW leads to more than 27% and 16% improvement in online *CTR* and *DTR*, compared to the predecessor [7], which uses Personalized PageRank[4] in discovering target users.

In what follows, we present the full details of Huawei *PSP*. We first overview *PSP* in Section 2. Then we give the work flow of the Application Layer and presents the motivation, principle and implementation of A-PARW in Section 3. After that, we apply our system on several real marketing tasks in Huawei App Store, and carry out detailed off-line and on-line evaluation in Section 4. Finally, we discuss some related works in Section 5 and conclude the paper in Section 6.

2 Platform Overview

The architecture of Huawei Push Service Platform (PSP) is shown in Figure 2, and includes *Distributed Storage Layer*, *Application Layer*, and *Evaluation Layer*.

The Distributed Storage Layer maintains two database systems for historical data storage and on-line caching. The HDFS (short for Hadoop Distributed File System) stores historical data, including users' download, click, and payment log data, which is the source data for our User-App bipartite graph (discussed later). The HBase (short for Hadoop Database) caches on-line data and users' feedback, which is critical for on-line monitoring and algorithm evaluation, and updates the historical data in HDFS periodically. In addition, this layer incorporates a Hadoop cluster to store large-scale datasets and provides parallel data processing.

The Application Layer consist of the major components (i.e., off-line target users mining and on-line pushing) of the platform. For different demands in practice, the Application Layer supports different Computing Engines, including

[4] xRank, proposed in [7], is exactly Personalized PageRank (PPR) and is equivalent to the D mode of PARW. More details are presented in [3].

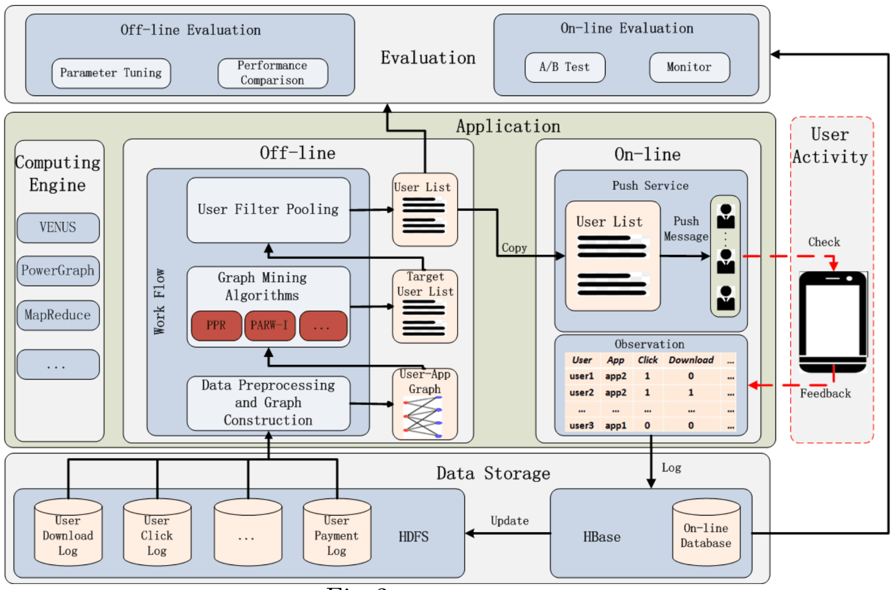


Fig. 2: PSP Architecture

graph engine and distributed computing engine. We will give more details of the Application Layer in Section 3 as it is the most challenging and important constituent of our platform.

The Evaluation Layer evaluates both off-line and on-line results. Off-line evaluation compares pre-defined off-line metrics of the results by different algorithms, which helps us to tune the parameters of the algorithms. On-line evaluation, such as A/B test, compares the performance of the algorithms which are carefully selected by off-line evaluation. The details of evaluation methods and metrics will be presented in Section 4.1.

3 Application Layer

The Application Layer of our PSP can be described as the following work flow:

$$History\ Data \xrightarrow[Graph\ Construction]{Pre-processing} User\text{-}APP\ Graph \xrightarrow{Graph\ Mining} Target\ User\ List \xrightarrow{User\ Filtering} User\ List \xrightarrow[Observation]{On-line\ Pushing} User\ Feedback \xrightarrow{Logging} Online\ Log \xrightarrow{Updating\ pediodly} Historical\ Data.$$

The input of PSP is a topic push activity, which is denoted as *seed Apps*[5] under a certain topic, such as music fans, cook lovers, etc. As an initial step, we provide *Data Preprocessing and Graph Construction* operation (Section 3.1) to generate User-App graph from users' download/click/payment historical data,

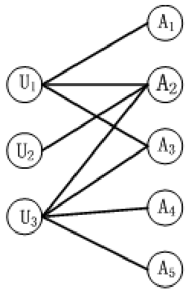
[5] Seed Apps are a small set of manually-labeled Apps.

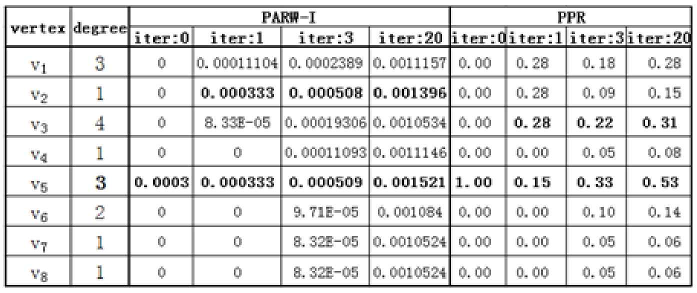
which is stored in HDFS. Based on this graph, we can mine target user group through *A-PARW* (Section 3.2). Then, according to some domain knowledge and rules, *User Filtering* filters irrelevant users to obtain the final user list (Section 3.3) and the module of *Push Service* (Section 3.4) sends the message to the selected users. After that, PSP will cache the users' feedback data and update these data to History data periodically. We will review various computing engines in supporting needs in this layer in Section 3.5.

In the rest of this section, we introduce the details of Application Layer according to the work flow briefly described above.

3.1 Data Preprocessing and Modeling

On server log data, the first step in the Application Layer includes two modules, Data Preprocessing and Graph Construction.
Data Preprocessing: PSP can set a series of rules according to demands, such as removing pre-installed or very popular Apps from raw data before graph construction, because installing such Apps will not reflect users' interests.
Graph Construction: In this module, PSP constructs an undirected graph $\mathcal{G} = (\mathcal{U}, \mathcal{A}, \xi)$ based on the preprocessed data, where $\mathcal{U}$ denotes the set of *users* vertices, $\mathcal{A}$ is the set of *Apps* vertices, and ξ is the set of edges. Since we only use historical information to record the interaction between users and Apps, the constructed graph $\mathcal{G}$ is a bipartite graph. For instance, in Figure 3, User vertices are on the left-hand-side and App vertices are on the right-hand-side. There exists an edge connecting U_i and A_i if user U_i installs App A_i. For example from Figure 3, U_1 installs three Apps A_1, A_2 and A_3.





vertex	degree	PARW-I				PPR			
		iter:0	iter:1	iter:3	iter:20	iter:0	iter:1	iter:3	iter:20
v_1	3	0	0.00011104	0.0002389	0.0011157	0.00	0.28	0.18	0.28
v_2	1	0	**0.000333**	**0.000508**	**0.001396**	0.00	0.28	0.09	0.15
v_3	4	0	8.33E-05	0.00019306	0.0010534	0.00	**0.28**	**0.22**	**0.31**
v_4	1	0	0	0.00011093	0.0011146	0.00	0.00	0.05	0.08
v_5	**3**	**0.0003**	**0.000333**	**0.000509**	**0.001521**	**1.00**	**0.15**	**0.33**	**0.53**
v_6	2	0	0	9.71E-05	0.001084	0.00	0.00	0.10	0.14
v_7	1	0	0	8.32E-05	0.0010524	0.00	0.00	0.05	0.06
v_8	1	0	0	8.32E-05	0.0010524	0.00	0.00	0.05	0.06

Fig. 3: An example User-App bipartite graph and A-PARW-I vs PPR running case

In addition, we assign uniform IDs to vertices in $\mathcal{A}$ and $\mathcal{U}$. More specifically, vertices in $\mathcal{U}$ are assigned with IDs from 1 to $|\mathcal{U}|$, and vertices in $\mathcal{A}$ are assigned from $|\mathcal{U}| + 1$ to $|\mathcal{U}| + |\mathcal{A}|$. For simplicity, we use v_i to denote vertex with ID i. For example, we use v_1 to v_3 to denote U_1 to U_3 and v_4 to v_8 to denote A_1 to A_5 respectively. We denote the adjacency matrix of $\mathcal{G}$ as $\mathcal{W}$, let $\mathcal{D} = diag(d_1, d_2, ..., d_N)$ with $d_i = \sum_j w_{ij}$ as the degree of vertex i, and define the Laplacian of $\mathcal{G}$ as $\mathcal{L} = \mathcal{D} - \mathcal{W}$.

In some push scenarios, such as online news recommendation, the graph has to be updated frequently as the hot spots are changing at any time. While in some other scenarios, such as the recommendation in application market like the one considered in this paper, the graph does not need to be updated so frequently because the interest of a user is unlikely to vary much from time to time. Hence, in our application scenario, we re-construct graph weekly with the most up-to-date information, which usually takes a few hours.

3.2 User Discovery via Graph Mining

After preprocessing, the most significant task for the push service is the problem of target user discovery. In this subsection, we carefully compare different choices.

As it is expensive to manually label all Apps and the result has no ranking information, the predecessor of PSP [7] applied PPR to mine target user group from some *seed Apps*. However, in our practice, PPR favors active users with high degree, who will download Apps with high probability no matter receiving push messages or not, and is likely to ignore the majority of inactive and semi-active users. Therefore, we need an algorithm that can mine relevant inactive and semi-active users. Below, we present the approximate Partially Absorbing Random Walk algorithm, *A-PARW*, which we found quite effective in mining target users from the seed Apps and has been adopted in our PSP push platform for Huawei App Store. (Due to space limit, the details are presented in [3].)

A-PARW: To mine target user group from a small number (e.g., 10) of *seed Apps*, we propose A-PARW by extending Partially Absorbing Random Walk (PARW) [13] for billion scale problems encountered in Huawei App Store. The formulation of PARW is $R^{\top} = (In)^{\top} \cdot (\Lambda + \mathcal{L})^{-1} \cdot \Lambda$, where R is the rank score vector, $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_N)$ is a diagonal matrix with $\lambda_1, \lambda_2, ..., \lambda_N$ being arbitrary non-negative numbers, and In is a vector of $In(v)$ with $In(v) = 1/|\textit{seed Apps}|$ if $v \in \textit{seed Apps}$ and 0 otherwise.

In PARW, a random walk is absorbed at state i with probability p_i, and is transferred via a random edge of state i with probability $1-p_i$. It is proved in [13] that a random walk starting from a set of low conductance vertices (referred as SP) is most likely absorbed in SP if $\Lambda = \alpha \cdot I$ (PARW-I), I is an identity matrix and α is a small positive value. One property of PARW-I is that the absorption probability varies slowly within SP, and drops sharply outside SP. This property suggests that PARW-I can effectively capture the underlying community structure of the graph.

As Figure 3 shows, low degree yet relevant vertex v_2 (the user represented by v_2 only installs A_2 (vertex v_5), which means he is more interested in A_2) absorbs higher score than v_3, which has high degree but with no deterministic preference, for A-PARW-I. In contrast, v_2 could not get good score for PPR. (Due to space limit, the details are presented in [3]).

However, to the best of our knowledge, there is no scalable implementation of PARW. Therefore we propose A-PARW in Algorithm 1 motivated by [1]. Our algorithm maintains a pair of vectors run and dry, starting with $dry = \mathbf{0}$ and $run = In$ (Line 1), then applies a series of push operations which transfer probabilities from run to dry while keeping no transfer out of dry. At vertex v_i, a push operation transfers $\lambda_i/(\lambda_i + d_i)$ fraction of run_i to dry_i (Line 3), then evenly

Algorithm 1 dry=A-PARW(s, Λ, γ) ▷ APPROXIMATE ALGORITHM OF PARW

```
Input: S: seeds, $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$: regularization parameter, $\gamma$: tolerance threshold
Output: A-PARW vector $dry$
 1: Initialize $dry = 0$ and $run = \{(s, 1/|S|)\}$
 2: while $run$ is not empty do
 3:     pop a queue $run$ element $(i, w)$ and $dry_i = dry_i + \frac{\lambda_i}{\lambda_i + d_i} \cdot w$
 4:     if $w > \gamma \cdot d_i$ then
 5:         for all links $(i, j) \in \xi$ do
 6:             if pair $(j, s) \in run$ then
 7:                 $s = s + \frac{w}{\lambda_i + d_i}$
 8:             else
 9:                 add a new pair $(j, \frac{w}{\lambda_i + d_i})$ to $run$
10:             end if
11:         end for
12:     end if
13: end while
```

distributes the remaining $d_i/(\lambda_i + d_i)$ fraction of run_i to v_i's neighbours (Line 5-11). We can control the precision through a strategy that A-PARW performs push operations only when $run_i \geq \gamma \cdot d_i$ (Line 4). As a result, we set γ to be 10^{-8} and select a limited number of iterations (i.e., 20) as A-PARW-I's stop condition. It is demonstrated to be good enough in our scenario.

3.3 Filtering Rule

In pratice, the target user list, mined through A-PARW, includes some users who are not suitable to send push messages. Therefore we can define some practical filtering rules to filter them out. For instance, we should not select users who have turned off the function of receiving push messages, or we may not want to send messages to the users who visit Huawei App Store every day, etc.

3.4 On-line Pushing

First of all, the module of *Push Service* will deliver messages as an alert on the notification center to the selected users' phone, whose push service is enabled and which is connected to the Internet. After that, these users will receive messages in their phone as shown in Figure 1a, and may choose to neglect it or click on. After clicking this message, users will enter the specific page of Huawei App Store or even download the Apps contained in this page. For example, when a music fan receives an alert about music Apps on her phone's notification center (e.g., as the second message shown in Figure 1a), she takes a look at music Apps in the display page (e.g., as in Figure 1c), after clicking the alert message. In the end, we utilize user's feedback to generate market strategies and filtering rules. For example, we are more likely to send a message of a music-like activity to a user who has clicked the music push activity before, and we are less likely to send a push message to a user who has never clicked any push message before.

3.5 Computing Engine

As presented in Section 3.1 to Section 3.4, there are two kinds of computing tasks in the Application Layer, raw data extraction and graph mining. The former is easy to parallelize while the latter is diffcult due to the heavy dependencies between vertices in a graph. Therefore, we choose MapReduce as general computing engine, but use graph engines, including VENUS [10] and PowerGraph [2], for graph mining. Specifically, we use a disk-based system–VENUS when push activity is not so urgent and the memory is limited; and we choose a memory-based distributed system–PowerGraph when push activity is highly urgent and resource is enough.

Table 1: Running Time (in seconds) of A-PARW-I on PowerGraph and VENUS

No. of push activities	1	10	100
PowerGraph	6.79	13.19	46.69
VENUS	2307.00	22588.90	N.A.

In order to compare the efficiency of VENUS and PowerGraph, we ran experiments on twitter-graph [8], which contains 41,652,230 vertices and 1,468,364,884 edges. To compare fairly, the experiments are conducted on the same machine and the parameters are set to be the same as stated in the previous section. As Table 1 presents, PowerGraph needs only 6.79 seconds to process one push activity, while VENUS needs around 40 minutes. For 10 and 100 push tasks, PowerGraph uses around 13 seconds and less than 50 seconds respectively. While VENUS has to run more than 6 hours when pushing 10 tasks. We don't test the case of pushing 100 tasks on VENUS as it needs several days to get the precise timing. But it's easy to estimate the time since it runs these tasks one by one.

4 Experimental Results

In this section, we first describe the data sets and evaluation metrics that are used in our experiments. Then, we compare the experimental results of A-PARW-I and PPR on real data set. Moreover, the details of experimental results on public data set are presented in [3] due to the space limit.

4.1 Data Set Description and Experiment Setting

We evaluated A-PARW algorithms on two data sets, MovieLens [6] and *APPData*, where *APPData* is collected from Huawei App Store. The difference between A-PARW-I and PPR is verified on both data sets, while evaluation on the real-life data sets furthermore confirmed the remarkable effectiveness of A-PARW-I.

Real-Life Dataset and Experiment Setting We performed experiments on real data set–*APPData*. It is the complete user downloading log from 2015/03/01 to 2015/08/31 in Huawei App Store, and includes 96,324,654 users, 487,649 Apps, and 1,778,160,959 edges.

We first conducted experiments to evaluate the effectiveness of A-PARW-I, and then we verify the property of A-PARW-I and PPR. Our experiments on *APPData* include both off-line and on-line evaluation.

[6] http://grouplens.org/datasets/movielens/

For off-line evaluation purpose, we collected users' feedback of nine push activities from Huawei App Store, for which the selected user lists were generated by PPR. The nine push activities are: **1:music; 2:camera; 3:instrument; 4:ticket; 5:listen book; 6:travel; 7:goodnight; 8:read; 9:internet**. We referred the users' feedback as the ground truth to compare the effectiveness of PPR and A-PARW-I. After receiving a push message of Apps, an interested user may click on it or even download this Apps. Therefore, we can distinguish the cases of click (or download) as follows: we refer a sample that the user clicked (or downloaded) the push message as positive, and the opposite case as negative. We ran PPR and A-PARW-I on *APPData*, and got top 1 million users as well as their ranking scores, respectively. In the off-line experiment, we adopt AUC as the evaluation metric.

In the on-line evaluation, we sent push messages to the same number of top users ranked by A-PARW-I and PPR, respectively. Making sure to receive the feedback from the majority of the users after two days, we compared the number of users who clicked (or downloaded) the recommended Apps, across the two sets of users picked by the two algorithms. We use CTR (click-through ratio) [7] and also DTR (download-through ratio) as the online evaluation metrics, where $CTR = |Users\ who\ clicked\ advertised\ app|/|Users\ who\ received\ ads|$ and $DTR = |Users\ who\ downloaded\ advertised\ apps|/|Users\ who\ received\ ads|$.

Public Dataset As we discussed in Section 3, A-PARW-I is able to identify more semi-active users than PPR because it capitalizes on community structure. In order to verify this property, we ran PPR, A-PARW-I on MovieLens data set and compared degree trends of their ranking lists. The results implied PPR favors high degree vertices and ignores the semi-active users. In contrast, A-PARW-I can discover semi-active users. Due to space limit, the details are presented in [3].

4.2 Evaluation on Real-Life Data set

Off-line Evaluation In off-line experiment, we used AUC to compare the ranking accuracy of A-PARW-I's and PPR's result list (we get top 1 million users from their result list). Table 2 presents click and download AUC improvement of A-PARW-I over PPR on the 9 push activities. As we can see, the performance of A-PARW-I is better than PPR at all the 9 push activities, on both click and download cases. Moreover, the improvement of A-PARW-I over PPR is more significant on download case. The improvement of A-PARW-I comes from the fact that A-PARW-I pays more attention to graph community information while PPR tends to high degree nodes. It means that A-PARW-I could find more semi-active user-nodes, which are of low degree but highly relevant to push activities.

On-line Evaluation In this subsection, we performed A-PARW-I and PPR algorithms on two different on-line push activities through Huawei's push service. Two lists of users are obtained by the two algorithms respectively, and the activity messages are push to these users [7]. Interested users may click the message to see the details of the Apps or perform further actions.

We calculated CTR and DTR from the log data of user feedbacks. For comparison, we define $CTR+$ and $DTR+$ as the improvement of A-PARW-I over PPR. As we can see in Table 3, the performance of A-PARW-I is significantly higher than PPR for both click and download on the two online push activities.

Table 2: Off-line improvement of A-PARW-I over PPR in 9 different push activities

	1	2	3	4	5	6	7	8	9
Click AUC+	6.5%	10.9%	4.8%	3.1%	6.0%	37.7%	31.1%	39.9%	11.8%
Download AUC+	7.6%	10.9%	7.9%	9.2%	3.4%	12.0%	8.3%	12.4%	5.4%

Table 3: On-line improvement of A-PARW-I over PPR

	ticket	music
$CTR+$	27%	82%
$DTR+$	16%	85%

Property of PARW In order to analyse the property of A-PARW-I and PPR, we study the tendency of CTR/DTR and the degree of the user vertices selected by the two algorithms and sorted by their scores.

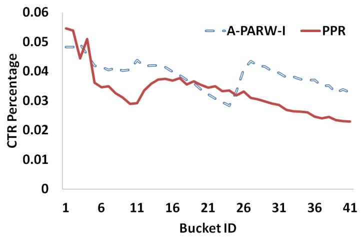
Tendency of CTR/DTR in sorted-steady-distribution. Figure 4a (4b) and Figure 4c (4d) represent tendency of activity ticket's (music's) CTR and DTR. In the figures, x-axis is the bucket identifier (each bucket includes 100,000 users) and y-axis is the CTR (respectively DTR) of the buckets. As we see, the curves of PPR and A-PARW-I have high CTR and DTR value at the beginning. However, PPR's curve drops more dramatically than A-PARW-I's; moreover, PPR's CTR and DTR sometimes increase at the tail of the curve. So A-PARW-I, which is steady and stable, selects the users who are more relevant to the push activity than PPR.

Change of degree in sorted-steady-distribution. Figure 5a (5b) presents the degree's tendency of A-PARW-I and PPR in the activity of ticket (music), where x-axis is same as Figure 4 and y-axis is the total degree of the users in the buckets. As we see, the tendency of both red and blue curves are similar across the two figures. The PPR's curve is higher than the A-PARW-I's at first but drops rapidly and A-PARW-I's curve is more smooth and steady. It can be concluded from these two figures that PPR prefers high degree nodes.

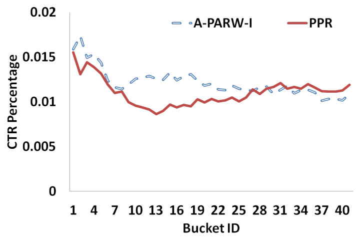
5 Related Work

Compared to pull-mode, which gives recommendations to user within item market, push-mode pushes specific messages to users according to their characteristics even when users are not in item market. So push recommendation is able to rebuild or strengthen the connection between item market and users.

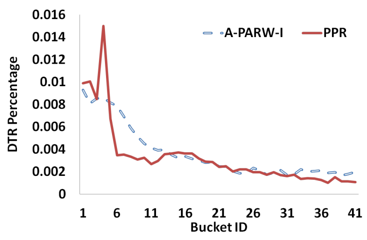
[7] Duplicated users in the two lists are only push the message once.



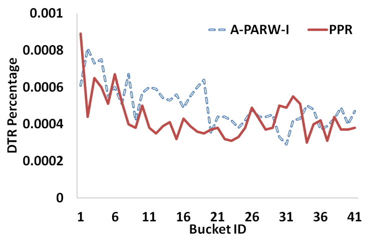
(a) CTR tendency of activity ticket



(b) CTR tendency of activity music



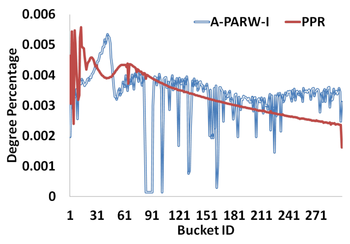
(c) DTR tendency of activity ticket



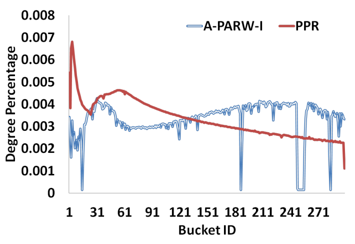
(d) DTR tendency of activity music

Fig. 4: Tendency of CTR/DTR

The crucial part of push-mode is target user group discovery, which can be solved by *rule-based* [5], *CF-based* [12] and *graph-based* [7,9] approaches. However, the rule-based methods can not take advantage of collaborative information among users, because it needs a set of rules that are defined so that the accuracy rate is low and not flexible enough. The CF-based methods tend to be ineffective in real-world scenarios because it requires a great deal of interactions of users and tags, which is problematic due to the sparsity of data. In graph-based approach, PageRank [11] is a well-known link analysis algorithm used by Google to rank websites according to their importance. There are many variants of PageRank, such as sensitive PageRank [6], xRank [7], and WTF [4]. However, PageRank based approach is biased to high-degree vertices. The authors of [13] propose a unified framework of graph mining and a new algorithm PARW-I, which can



(a) Ticket



(b) Music

Fig. 5: Tendency of users' degree

capture the community structure to overcome the weakness of PageRank. So we design approximate PARW, namely A-PARW, in our system.

6 Conclusions and Future Works

In this paper, we introduced Huawei Push Service Platform (PSP) to perform push recommendation by selecting target user group for a given push message. PSP includes potential users mining, online pushing, feedback caching and evaluation. In addition, we proposed A-PARW for target user group discovery on large scale data and presented a detail analysis among different choices of algorithms theoretically and empirically. We highlighted that A-PARW-I is able to discover the most relevant potential users and improve the performance of push service. As a live system, PSP supports the push recommendation in Huawei App Store and leads to a significant improvement over PPR.

Acknowledgement. This research was supported in part by NSFC under Grant No.61572158 and Shenzhen Science and Technology Program under Grant No.JCYJ20160330163900579. We thank Dr. Qin Liu, Dr. Junbo Zhang and Chenzi Zhang for the help in scaling up PARW. We also thank Dr. Zhenhua Dong, Zhirong Liu and Benwei Gong for the valuable discussion and feedback.

References

1. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: FOCS (2006)
2. Gonzalez, J.E., Low, Y., Gu, H., Bickson, D., Guestrin, C.: Powergraph: Distributed graph-parallel computation on natural graphs. In: OSDI (2012)
3. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: A graph-based push service platform. https://arxiv.org/abs/1611.09496 (2016)
4. Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., Zadeh, R.: Wtf: The who to follow service at twitter. In: WWW (2013)
5. Han, J., Pei, J., Kamber, M.: Data mining: concepts and techniques. Elsevier (2011)
6. Haveliwala, T.H.: Topic-sensitive pagerank. In: WWW (2002)
7. He, X., Dai, W., Cao, G., Tang, R., Yuan, M., Yang, Q.: Mining target users for online marketing based on app store data. In: IEEE International Conference on Big Data (2015)
8. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: WWW (2010)
9. Li, X., Wang, Y.Y., Acero, A.: Learning query intent from regularized click graphs. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (2008)
10. Liu, Q., Cheng, J., Li, Z., Lui, J.: VENUS: A System for Streamlined Graph Computation on a Single PC. TKDE (2016)
11. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. In: WWW (1999)
12. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in artificial intelligence (2009)
13. Wu, X.M., Li, Z., So, A.M., Wright, J., Chang, S.F.: Learning with partially absorbing random walks. In: NIPS (2012)