IEEE SYSTEMS JOURNAL, VOL. 11, NO. 1, MARCH 2017

Personalized Multimedia Recommendations for Cloud-Integrated Cyber-Physical Systems

Chi Harold Liu, *Member, IEEE*, Zhen Zhang, and Min Chen, *Senior Member, IEEE*

***Abstract*—Portable smart devices have paved the way for accessing and capturing different types of multimedia contents with human interactions, leading to the emergence of cyber-physical systems (CPSs). Although the massive data collected from these physical terminals can contribute to the improvement of their quality of lives by building smart communities, CPSs intensify the information overload problem. Therefore, plenty of research efforts have been paid to develop multimedia recommender systems. However, most existing research activities neglect its time-varying features due to system dynamics, i.e., not only the amount of input data constantly grows, but also the change of user behaviors and system operating environment. In order to sustain the high accuracy of recommendations, the system in a CPS has to be updated regularly. However, the more often the update proceeds, the more the cost of other computational resources. To this end, in this paper, we propose an adaptive recommender system by using feedback control frameworks in CPSs. The proposed solution continuously monitors its changes and estimates the loss of performance (in terms of accuracy) to overcome the data aging problem and justify if the current "revisiting ratio" between the new and old items can still accurately reflect current user behavior. Theoretical analysis and extensive results by using a real data set in a cloud setting are supplemented to show the advantages of the proposed system.**

***Index Terms*—Cyber-physical systems (CPSs), feedback control framework, recommender systems.**

I. Introduction

After the occurrence and development of the Internet, people's understanding about information has been enormously changed. In the past, the transmission of information often took a long time, and thus, the lack of information has once become a strong obstacle for communications. However, the availability of data centers and wide-area wireless networks have paved the way for accessing and capturing different types of multimedia contents (i.e., text, images, audios, and videos) [1]. As a result, people can acquire the latest information (e.g., news, friend's condition, and new movies) anywhere and anytime with portable smart devices like the smartphone and iPad [2]. This intimate coupling between the cyber and physical spaces has led to the emergence of cyber-physical systems (CPSs) [3], as a new generation of systems with computational and physical capabilities that can interact with humans through many new modalities [4]. Typical examples of CPSs are cyber-physical social networks (that build social networks with data collected sensors) and intelligent transportation systems to gather and process multimedia data transmitted from the vehicles.

Nevertheless, these physical things, i.e., smart devices, are usually associated with limited computational processing speed, memory size, etc., and in a CPS application, information overload has become a serious problem, particularly for these smart mobile devices that cannot store a lot of multimedia contents for users. The power of recommending interesting information to specific users has emerged as a bottleneck on the path to intelligent information processing in CPSs. Therefore, designing analytical and efficient recommendation models to address the information overload problem is very important [3]. In order to address this issue, recommender systems have been proposed as an essential tool for users to navigate the plethora of contents according to their own interests. Examples include the user interfaces provided by Amazon [5], Netflix [6], and Last.fm [7]. For Amazon, a digital enthusiast may find that the login page is filled with all sorts of recommended equipment which he/she may like. Also, a young mother will easily find a satisfying cookbook with the help of the "Guess you like" section. This is because Amazon has made full use of the user's purchase history to build a personalized recommendation model and then applies it to predict the user's behavior. This approach not only brings high profits but also significantly eases the shopping experience for customers. After the first inception of the recommender system, a number of engines (i.e., item-based recommendation, user-based recommendation, collaborative filtering (CF) algorithm [8], and singular value decomposition (SVD) algorithm) have blossomed.

However, there are still fundamental issues that need to be considered from theoretical analysis perspectives for recommender systems in CPSs. First, although a lot of schemes have been raised to improve the quality of results, they seldom consider the system loss (e.g., resource and accuracy) during the recommendation process, particularly in a big data environment. During the process of system operations, data accumulate continuously, which requires more and more time and computing resources for data transmission and model training. In the meantime, people often change their likes and dislikes as they are affected by seasonal trends or new things. Therefore, obsolete training sets also need to be cleaned up when they cannot correctly reflect a user's current behavior. In order to

Manuscript received February 8, 2015; revised May 5, 2015; accepted May 30, 2015. Date of publication June 19, 2015; date of current version March 10, 2017. This work was supported by the National Natural Science Foundation of China (Grant No. 61300179). *(Corresponding Author: C. H. Liu).*

C. H. Liu and Z. Zhang are with the School of Software, Beijing Institute of Technology, Beijing 100081, China (e-mail: chiliu@bit.edu.cn; zhenzhang@bit.edu.cn).

M. Chen is with the School of Computer Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: minchen2012@hust.edu.cn).

Digital Object Identifier 10.1109/JSYST.2015.2440431

sweep useless data in time, some systems set an update cycle and make regular updates. However, the change of system is not following a linear growth, and thus, a more practical solution should be proposed, instead of fixed update frequency. Second, existing recommender systems tend not to recommend "old" items that users have viewed before. They believe that users will not get interested to items that they have seen before. It may be true in some scenario, such as an online bookstore where customers barely choose books they already have. However, in some other situations like online video viewing, it is very likely that users will repeatedly watch videos they liked, particularly TV series. Therefore, it is necessary to consider the user differences and make recommendations according to their distinct characteristics.

Towards this end, in this paper, we aim to design an efficient feedback control framework to balance the stability and accuracy of the recommender system for CPS applications like smart communities while minimizing the cost of resources. Two feedback controllers (i.e., training set controller and recommendation list controller) are introduced to monitor the system performance and dynamically decide whether the benefit of performing an update exceeds the cost of resources. A novel concept of "revisiting ratio" is proposed to represent the percentage of new and old items, and a preference model is designed to monitor the current "revisiting ratio" of user profile and adjust the "revisiting ratio" of recommendation list when the change of user behavior is detected.

The contribution of this paper is fourfold.

1) We study the patterns of user behavior for a large-scale real data set, including their periodically viewing trend, preference on popular video channels, geographic locality characteristics, invariance, and dynamics.
2) We introduce a personalized recommender system with both historical and new items. A new concept of "revisiting ratio" is introduced to represent the percentage of new recommended items and old ones. New recommendations are made from the traditional recommender system, while old ones are produced from the newly proposed "preference model."
3) We design a training set controller to monitor the status of the system. The controller decides when and how to make an update to the training set, based on the Internet congestion control theory. We also propose the "slow recovery" and "fast adjustment" approaches to make the system accuracy recover fast to the steady state.
4) We propose a recommendation list controller to capture the change of user behaviors. By monitoring the trend of "revisiting ratio," this controller automatically adjusts the components of the recommendation list to fast fit on system dynamics. Based on the proportional-integral-derivative (PID) control theory, we propose a prediction model and use the ordinary least squares (OLS) regression method to perform the update.

The rest of this paper is organized as follows. In Section II, we highlight related research activities. Section IV establishes a formal model of our system. Section III introduces our observations on user behaviors from a real online video watching data set. Section V describes the designs of two control loops and the architecture of the entire feedback control framework. Extensive experimental results are presented in Section VI, followed by the conclusions and future work given in Section VII.

II. Related Work

CPSs have become a very interesting research area [9]. An increased dependence on CPSs led to the collection of a vast amount of multimedia data, which brings the information overload problem [10]. Therefore, plenty of research efforts have been paid recently on how to develop multimedia recommender system in CPSs. In [11], Yu *et al.* make use of location and trajectory data from GPS-enabled mobile devices to recommend geographically related friends in cyber-physical social network. Gao *et al.* build a cross-domain recommendation model for CPSs with an eye to tackle the data sparsity problem [3].

Since the quality of recommendations largely determines the merits of the system, recommendation algorithms become the key problem for both academics and industrial circles [12]. The CF-based recommendation is the earliest and is one of the most successful recommendation technologies, based on the assumption that users will not change their preferences over time. Generally, it first measures a user's distance by calculating the similarity between their preferences. Then, the K-nearest users are chosen to be neighbors of the target user [13]. Finally, the system predicts how much the target user is interested in a particular article according to his/her neighbors' attitudes. As pure CF recommendation process does not require any information about items, it can handle the recommended objects including unstructured complex objects such as music and movies. However, this approach is based on the historical data, i.e., it cannot get a good result to new users and items [14]. In addition, it also requires a lot of computation resources since user preferences are often stored in a sparse matrix [15]. Content-based recommendation [16] was originated in the field of information retrieval and information filtering, which makes use of the content information of the item. Different from CF, it does not require the user's evaluation of the item but requires the user's interests and preferences extracted from the description about the content by using machine learning techniques. Focusing on the features of an item, this approach can achieve high accuracy without large-scale users, and it overcomes the cold start problem. However, it requires good structural characteristics during the feature extraction, and thus, these features are often very difficult to obtain. Aside from these two algorithms, there is also a method called knowledge-based recommendation. Similar to the reasoning technology, it makes use of additional casual knowledge and requires strong interactions with users [17], [18].

Apart from the key recommendation algorithms, system diversity and dynamics are also the key features of recommender systems. Therefore, there are also a lot of approaches proposed to follow the trends of users and the change of item [19]. In [20], Jambor *et al.* summarize the reasons of temporal dynamics as the change of the user's preference and system itself.

Lathia *et al.* perform a user survey [21] to show that temporal diversity is an important facet of recommender systems. Burke [22] proposes a temporal leave-one-out approach to provide insight into the user-specific and system-level evolution of the recommendation behavior. Furthermore, Bakir and Albayrak in [23] notice that users' tastes also have diversity. To determine user preferences, they examine the dates when users ranked products and use this information to provide customized suggestions. In addition, The first dynamics web recommender system is proposed by Taghipour *et al.* in [24]. He also modeled the recommendation process as a Q-learning problem.

Unlike these approaches where statistical methods play an important role, the use of control theory makes update as a response to dynamics. Classical control theory was proposed as an interdisciplinary branch of engineering and mathematics and was designed to deal with the behavior of dynamical systems. In [25], Ogata provides a gradual development of this theory by investigating the analysis and design of discrete-time control systems. He discusses in detail the theoretical background and shows how to use MATLAB in system design. Parekh *et al.* [26] address that adding a controller that manipulates the target system's tuning parameters can achieve service level objectives for a software system, which is then applied in the email server to maintain a reference queue length. Meng *et al.*[27] evaluates a control theory-based rating recommendation updating algorithm in order to exclude unfair ratings from reputation systems. Bohang *et al.* [28] applies the control theory into the urban traffic adaptive control problem and design a closed-loop traffic control scheme. Control theory is also applied in supply-chain-management system, Agaran *et al.* model a supply chain system through state-space techniques and then design a linear-quadratic-Gaussian controller [29].

In summary, all existing research activities cannot fully balance the workload of computing and the accuracy of recommender systems when the multimedia data are accumulated from CPSs. Therefore, in this paper, we aim to propose a personalized recommender system for cloud-integrated CPSs.

III. Observations of User Behaviors From a Real Online Video Viewing Data Set

Specifically, in this paper, we consider a CPS application as video watching in smart communities, which supports a number of reliable recommendation services to improve the quality of lives. To illustrate this scenario, we use a real data set acquired from a large online video media platform, and we start from discussing the observed user behaviors in video watching to reveal the characteristics behind that.

A. Data Set Descriptions

Our data set is made up of 14 watching logs from a famous online video website in China that contains 1 000 000 users and 2000 videos in total. An average of 1 500 000 records per file is observed, and its size is around 8 GB each. Each record line has 48 attributes, including user identifier (ID), item ID, watching time, IP address, user location, channel information, etc., describing different aspects of the user's watching behavior.

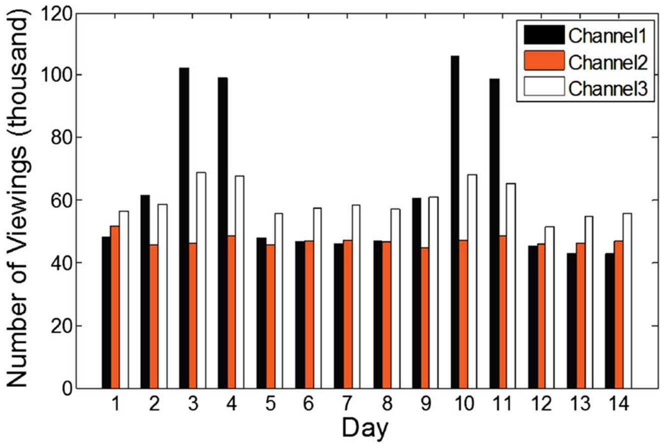

Fig. 1. Trend of viewing workload of each day by three different channels.

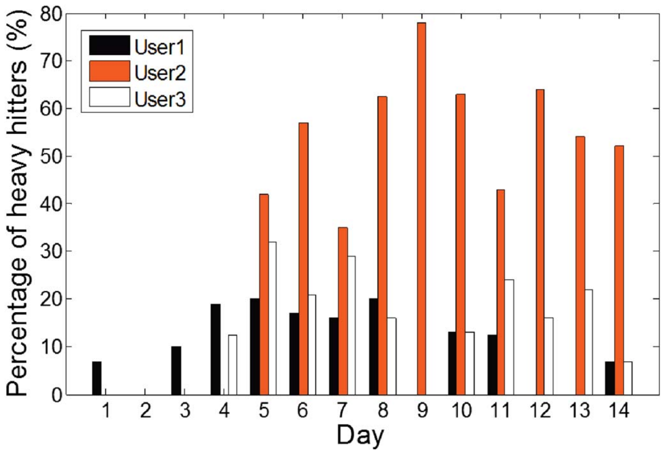

Fig. 2. Percentage of top 20 channels among all channels that three different users have watched on each day.

B. Periodic Viewing Trend

Figs. 1 shows the overall trend of viewing workload on each day for three different channels. It is obvious that channel 1 follows a clear periodic weekday/weekend trend over a week, and the number of viewings on weekends (days 3 and 4 and 10 and 11) is remarkably higher than that of workdays. Compared to channel 1, channel 3 attracts more viewings during the weekdays, but less on weekends. As for channel 2, its characteristics of viewing workloads on Saturday and Sunday are almost identical as that of the weekdays. This is highly likely due to the type of these three channels. Channel 1 is a TV series channel which is often updated on weekends, and thus, followers watch it a lot after it is updated (i.e., on weekends). On the contrary, channel 2 is a normal daily news channel which broadcasts every day. Since people may only focus on the news of the day, its viewing workload tends to be stable for the whole week. Channel 3 is a movie channel that contains all released films, and thus, users prefer to watch it when they have time at any time in a week. Therefore, its viewing workload is the highest, and not much difference can be observed on Saturdays and Sundays.

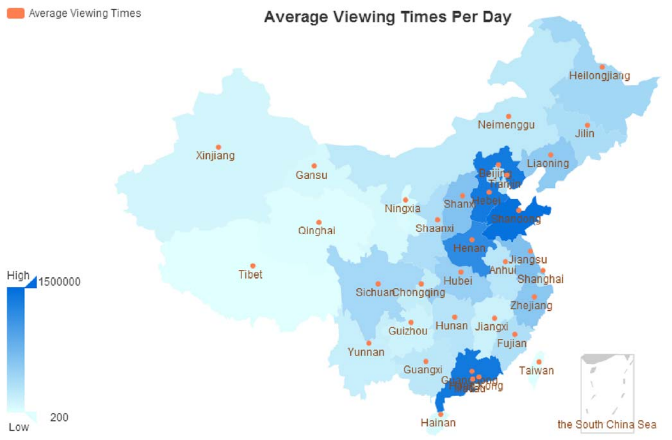

Fig. 3. Daily average of viewing times of each province in China. Darker area represents the larger geographical workload the users generate.

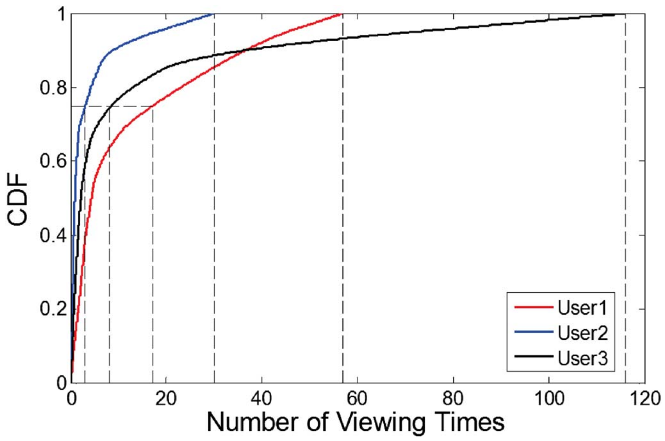

Fig. 4. Cumulative fraction of viewing times for different users.

C. Workload of Top Videos

Since the top ranked videos usually attract a great deal of viewings and consume much system resource, we select three different users and analyze the percentage of top 20 channels over all channels that they have watched on each day. Fig. 2 shows the diversity of a user's preference to these top 20 channels. We observe that, first, top channels occupy much in some user's profile (e.g., user 2). It is obvious that more than 30% of channels he/she watched on each day is top channels, and this ratio goes even up to 78% at day 9. On the contrary, user 1 only watches less than 20% of top videos, which indicates that he/she does not like to follow the current trend. Second, different users have different reactions upon the day's top channels. For example, user 2 likes the top channels of day 9 a lot, but the other two users have no interests on them. However, this situation is different at day 4 and day 12. In other words, not all these users have the same preferences over time. Therefore, a good recommender system design should fully consider this ratio (between the viewed top channels and all channels) as it can effectively reflect a user's preferences over time.

D. Geographical Locality

Fig. 3 describes the daily average of viewing times of each province in China. Darker area represents the larger geographic workload that users generate. We can easily find out that most users come from Beijing, Hebei, Shandong, and Guangdong provinces. In addition, users in different areas have different viewing behaviors.

E. Variance and Dynamics of User Preferences

We consider that the watching behavior of a user has two main characteristics: dynamic and invariant. That is, in a short period of time, user behaviors and interests are likely to be dynamic and will change. However, in the long run, a user's preference is invariant, particularly when they are following TV series. To clarify this observation, Fig. 4 illustrates the cumulative fraction of viewing times for three different users. We observe that the curves can be divided into two segments: low viewing part (0–20 times) occupying 75% of data and high viewing part (more than 20 times) occupying 20% of data, i.e., users' preferences can be very dynamic. Nevertheless, when we focus on the high viewing part, it is obvious that their behaviors are invariant since all three users have fixed channels to watch, particularly for user 3 who has watched one channel for 120 times.

IV. System Model

In this section, we first present a formal model for describing the system architecture and then introduce the system flow.

A. Assumptions and Notations

In a smart community application of CPSs, we consider that a recommender system is embedded in an online video application, where users holding their smart devices actively browse our recommendations and watch those they are interested in. Since the procedure of our proposed recommender system to make recommendations is identical for all users (however, the results are personalized), for simplicity reasons, we drop the user index for the following discussions. The website/application will store daily new log files $l(t) = \{l^i(t)\}$, where $l^i(t)$ means the ith log at day t, into a persistent database at the end of day. We also define a training set, denoted as $\underline{T}_m(t) = \{l(i)|i \in [m, t]\}$, where m is the starting point of the current training set and t denotes the current day. It can be regarded as a sliding window in time, within which all data are used to make recommendations. The final recommendations, denoted as $R_f(t)$, consist of the new recommendations $R_n(t)$ and old ones $R_o(t)$, or $R_f(t) = R_n(t) \bigcup R_o(t)$. $R_n(t)$ is produced by our recommendation module (see Section V-E), while $R_o(t)$ is the output from our proposed preference model (see Section V-C). Each day, the training set controller and recommendation list controller are running at background and monitoring the system performance. They take error signals as input, then use the control function to decide how much

TABLE I
LIST OF NOTATIONS AND DESCRIPTIONS

Notation	Explanation
$r_i(t)$	Watching record to item i at day t
$l(t)$	Log file at day t
$u(t)$	User profile at day t
$\underline{T}_m(t)$	Training set window from day m to day t
$\kappa(t)$	Accuracy parameter of training set controller at day t
k_0	Reference performance of training set controller at day t
$\epsilon(t)$	Error signal of training set controller
$\mu(t)$	Control signal of training set controller at day t
$\lambda(t)$	"Revisiting ratio" of recommendation list at day t
$\varphi(t)$	"Revisiting ratio" of user profile at day t
$\varsigma(t)$	Error signal of recommendation list at day t
$R_n(t)$	New recommendations at day t
$R_o(t)$	Old recommendations at day t
$R_f(t)$	Final recommendations at day t
$\zeta_i(\mathcal{A})$	Number of clicks on channel i in area $\mathcal{A}$. Specifically, when $\mathcal{A} =$ all indicates the entire area the data set covers, and when $i =$ top indicates the number of clicks on top channels
G_i	Geographical preference index of channel i
G_i	Geographical preference index of channel i
T_i	Percentage of top channels of channel i
D_i	User tolerance on channel i
wp_i	User dragging times of channel i

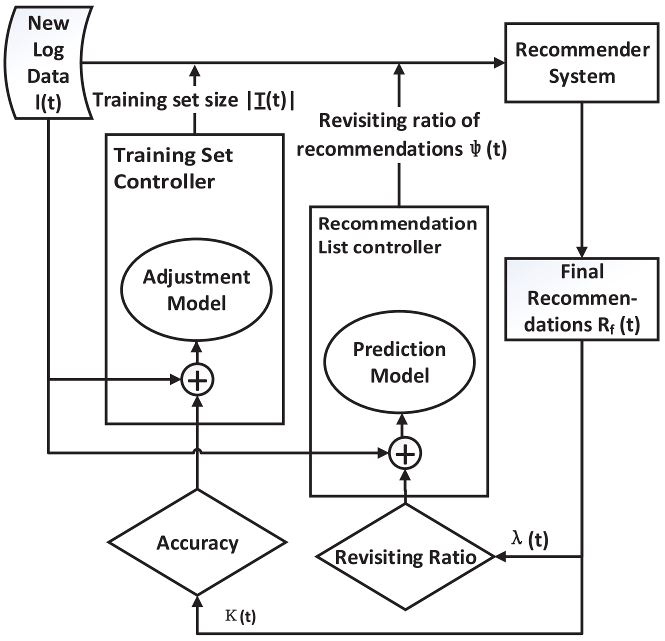

Fig. 5. Closed-loop control of our proposed recommender system, including a training set controller and a recommendation list controller.

adjustment should be made to their controlled modules, and then retrain the system model at background when necessary. For the training set controller, it continuously monitors the performance variable $\kappa(t)$ (including recall/precision value), compares with the predetermined reference value k_0, and computes the error signal $\epsilon(t)$. Then, it decides whether m should be updated and how should it be adjusted (i.e., to expand or reduce). For the recommendation list controller, it first compares the "revisiting ratio" of recommendation list $\lambda(t)$ and the "revisiting ratio" of user profile $\varphi(t)$ (see Section V-E) and then generates $\lambda(t+1)$ from error signal $\varsigma(t)$. Under the self-control of these two controllers, our solution can provide high-accuracy recommendations with low cost every time users log in the website/application. Table I shows the list of notations used in this paper.

B. System Flow

The system flow is shown in Fig. 5 and can be summarized as follows.

Step-1: We take $l(t)$ as input and perform data preprocessing, including data cleaning (to eliminate noises and then extract useful fields by data reduction), and training data set adjustment, as $\underline{T}_m(t) = l(t) \bigcup \underline{T}_m(t-1)$.

Step-2: The recommender system module produces $R_n(t)$ and $R_o(t)$ by using the proposed preference model. Final recommendations become $R_f(t) = R_n(t) \bigcup R_o(t)$.

Step-3: When the proposed two controllers receive a user's feedback $u(t+1)$ at the next day $t+1$, they starts to evaluate the current system performance. For the training set controller, it calculates $\kappa(t)$ and $\epsilon(t)$, takes $\epsilon(t)$ as input, and then applies the proposed adjusting strategy to adjust m. For the recommendation list controller, it evaluates system performance by comparing $\lambda(t)$ and $\varphi(t)$. The comparison result $\varsigma(t)$ is then sent to generate $\lambda(t+1)$ by using the proposed prediction model (see Section V-E).

Step-4: The aforementioned process will be repeated each day.

V. Proposed Feedback Control-Based Recommender System for CPSs

In this section, we introduce how we design a novel feedback control-based recommender system. We first explain the consumption of our implicit rating; then, we introduce an open source recommender system (denoted as recommendation module) for new items. After that, the mechanism for recommending old items by using the proposed preference model will be described. Finally, we introduce two feedback controllers and explain their control strategies.

A. Implicit Rating

The use of explicit rating ranges from grading students' homework to assessing consumer goods. A central feature of explicit ratings is that the evaluator has to examine the item and assign a value to it based on the rating scale [30]. However, this value is usually very hard to obtain, and manual annotation often imposes a cognitive cost. These problems have led to speculations that implicit ratings may be a solution [31].

Implicit ratings include measures of interests such as whether the user read an article or not, and if so, how much time the user has spent on reading it. There are several types of implicit data that can, in principle, be captured and studied. The authors in [32] use three types of implicit data: read/ignored, saved/deleted, and replied/not replied. In [33], Morita and Shinoda use reading durations in place of the read/ignore attribute.

Since our data set lacks of explicit ratings but has some other related information that can be used to indicate a user's preferences, we decide to pick some representative ones and integrate them to make implicit ratings by using the ordered weighted averaging (OWA) [34] operator. We first explain these four parameters that we consider, and then, we describe how we adopt OWA to make the ratings.

1) Fraction of Clicks on a Channel: The first factor is the percentage of clicks on a particular channel i among the overall number of clicks for all channels, as a normalized fraction. Let the number of clicks on channel i in area $\mathcal{A}$ be denoted as $\zeta_i(\mathcal{A})$. Then, $\mathcal{A} =$ all indicates the entire area that the data set covers, and $i =$ top indicates the number of clicks on top channels. We have

$$C_i = \frac{\zeta_i(\text{all})}{\sum_i \zeta_i(\text{all})}. \tag{1}$$

The higher this value is, the more the user prefers this channel.

2) Geographic Locality Index for a Channel: As shown in Fig. 3, different areas have geographic diversity, but users in the same province may share the same preferences. Thus, we consider the popularity of clicking number in an area p among the entire number of clicks across all areas for a particular channel i as

$$G_i = \frac{\zeta_i(p)}{\zeta_i(\text{all})}. \tag{2}$$

For instance, if a user u locates in Beijing, then his/her geographic preference to channel i can be calculated by $G_i = \zeta_i(\text{Beijing})/\zeta_i(\text{all})$.

3) Percentage of Top Channels: As mentioned earlier, users have different preferences to top ranked channels. In other words, considering the percentage of top channels in a user profile can be very helpful to predict a user's viewing behavior. Therefore, we use T_i to denote the percentage of the clicking number for channel i among all clicks of top channels as

$$T_i = \frac{\zeta_i(\text{all})}{\zeta_{\text{top}}(\text{all})}. \tag{3}$$

4) User Tolerance on a Channel: Generally speaking, when a user gets bored with the content of the current video, he/she tends to drag the progress bar and make it fast forward. Therefore, the more drags a user has performed, the less the preference that he/she may have to this channel. To normalize this parameter to between 0 and 1, we adopt the reciprocal form of this value as the user tolerance on channel i as

$$D_i = \frac{1}{\wp_i} \tag{4}$$

where $\wp_i$ denotes the dragging times of the channel i, which can be extracted from our data set.

5) OWA Operator: The OWA operator is one of the advanced fusion operators that maps a vector (size n) of different values to a single fused value by using a normalized weight vector [35]. This fusion operator actually has the capability of spanning the whole averaging operator domains [36]. Here, the aforementioned four considered parameters (i.e., fraction of clicks on a channel, geographic locality, percentage of top ranked channels, and user tolerance on a channel) are denoted as C_i, G_i, T_i, D_i. Assume that function $F(i)$ denotes the implicit rating to channel i, with weighting vector $\underline{w}$, where $\underline{w} = \{w_1, w_2, w_3, w_4\}$ and $\sum_{j=1}^{n} w_j = 1$. We have

$$F(i) = \sum_{j=1}^{n} w_j b_j \tag{5}$$

where b_j equals to the jth biggest element in the bag $< C_i, G_i, T_i, D_i >$. The bigger the value, the more the users may like this item from this perspective. We therefore order these four parameters in the descending order and use the exponential OWA operator to find out the weights of each vector as $w_1 = \eta$, $w_2 = \eta(1-\eta)$, $w_3 = \eta(1-\eta)^2$, and $w_4 = (1-\eta)^3$, where parameter η belongs to the unit interval $0 \le \eta \le 1$.

B. Open Source Recommender System by Using Myrrix

In our design, we leverage an open source recommender system, called "Myrrix" [37], to make *new* recommendations. It is powered by Apache Mahout and proposed to build a temporal high-scalability recommendation engine in big data environment. There are two layers in this system. By using the machine learning model updated by the computation layer, the serving layer can answer requests, records input, and provides recommendations in real time. Therefore, we believe that the fundamentals of Myrrix serve as an ideal recommender system due to its simplicity, scalability, and universality.

C. Preference Model

For a certain item i, we consider that there are three aspects that are key to decide a particular user's preference on i. First, the more frequent the user views an item, the more he/she may like it. Thus, historical viewing times $\ell(t)$ is of our concern to the system design. Second, we consider this user's last viewing of the item $\jmath(t) = t - n(t)$, where $n(t)$ is the day of the last viewing record. If i has not been watched for a relatively long period of time, one may conclude that the user highly likely has already lost interests of it. Finally, we introduce a new metric called "loyalty value," denoted as $\Psi(t)$, to uniquely represent the *cyclical* viewing behavior of a user. That is, if an item has been viewed every Saturday, it is more likely to be viewed again next Saturday. This always happens for TV series that has predefined showing time. The loyalty value can be computed by two variables, the number of intervals $\epsilon(t)$ and loyalty times $\varpi(t)$, as

$$\Psi(t) = \frac{\varpi(t)}{\epsilon(t)}. \tag{6}$$

An example to calculate the loyalty value is shown in Fig. 6. First, we pick the days when i has been watched by u. Second, we number the intervals of neighboring two days and mark the

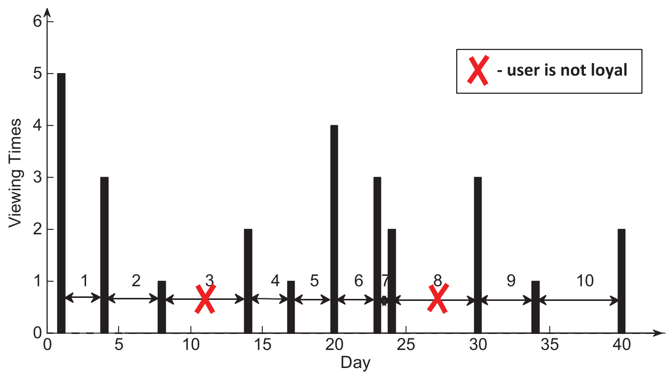

Fig. 6. Example to calculate loyalty value. In this scenario, ten intervals are observed in total, and the user is not "loyal" in the third and eighth days, marked by a red cross. Therefore, $\varpi(t) = 8$.

ones lower than the loyalty cycle (i.e., 7 days) as $\varpi(t)$. Third, we sum up the overall number of intervals as $\epsilon(t)$. Finally, we calculate the loyalty value as shown in (6). As shown in the figure, we randomly select a video watching log of a user and arbitrarily set the loyalty cycle to five days. It is clear that there are $\epsilon(t) = 10$ and $\varpi(t) = 8$ in this example. Therefore, the loyalty value of this user to this particular video is 0.8, which shows a high degree of user loyalty to this item.

Then, to consider the historical viewing times $\ell(t)$, the duration from used training set $\jmath(t)$, and loyalty value $\Psi(t)$, we compute the degree of user preference $\Theta(t)$ to an item as

$$\Theta(t) = a \times \Psi(t) + b \times \ell(t) + c \times \jmath(t) + d \tag{7}$$

where a, b, c, d are four parameters derived by logistic regression. The regression coefficients are estimated by using maximum likelihood estimation. An iterative process is used to find a closed-form expression for the coefficient values that maximize the likelihood function. This process begins with a tentative solution, revises it slightly, and repeats this revision until an improvement is minimum. At this point, the process is said to have converged, and the parameters are finally set.

D. *Training Set Controller*

In this closed-loop controller, we focus on the adjustment of the size of the training set by investigating its impact on the system accuracy. When new users join the system or new items are added to the website as just released, the overall volume of data increases over time. Meanwhile, existing users also tend to change their interests, which further complicates the scenario. To this end, we explicitly consider the data aging and data deficient problems that may result in inaccurate recommendations. The former happens when we build the training set with outdated data. Since these data cannot reflect a user's current preference anymore, our recommendations can be quite inaccurate. The latter occurs when the current data set does not contain enough user information to identify the user pattern. In addition, training a recommendation model is very time consuming and thus cannot be easily repeated [38].

To solve the aforementioned problems, we propose a training set controller. In its control loop, we use the achieved "precision" $\kappa(t)$ as a metric to evaluate the system perfor mance. We also set a reference value, denoted as κ_0, to represent our expected system performance. By comparing these two values, an error signal $\epsilon(t)$ is generated to decide whether to perform an update on the current training set. If so, we use the adjustment model to update m iteratively, until the accuracy of recommendations achieves the optimum. The detailed procedure is described as follows.

Step 1: System performance evaluation: We take $l(t+1)$ as a user's feedback to our recommendations and compute $\kappa(t)$ as

$$\kappa(t) = \frac{|R_f(t) \cap l(t+1)|}{|R_f(t)|} \tag{8}$$

where $|R_f(t)|$ denotes the size of $R_f(t)$.

Step 2: Calculate the error signal as: $\epsilon(t) = \kappa_0 - \kappa(t)$.

Step 3: Generate the control signal by the proposed adjustment model: Our proposed adjustment model aims to use $\epsilon(t)$ to generate control signal $\mu(t)$. Our adjustment model is derived from the well-known network congestion control theory in transmission control protocol (TCP)/IP, where "slow start" and "congestion avoidance" are used.

Of the first two phrases of TCP/IP congestion avoidance algorithms, the former applies to the start stage where too much data packets tend to be sent into the network space. In order to avoid congestion, we ask the host to stop and wait for a response every time after sending a packet. If a reply is successfully received, we increase the maximum segment size by one, which makes the value of congestion window (Cwnd) increase exponentially with the round-trip time to maximize the use of network bandwidth resources. When congestion window (Cwnd) exceeds the "slow start threshold" (SSthresh) that we set, the "slow start" process is switched to the "congestion avoidance" phrase where Cwnd changes to linearly increase. With the switch of these two phrases, we can successfully avoid excessive growth of network and maximize the network throughput in the same time [39].

Slightly different from these, we propose "slow recovery" and "fast adjustment" approaches to optimally adjust m to the best scope. When $\epsilon(t) > \kappa_0$, we start our iterative updating process until the recommendations made by the current training set can achieve the expected κ_0. During this process, recommender systems will serve as a tool to inspect the effect of the update. Similar to "slow start threshold" (SSthresh) in TCP, we introduce a novel concept of "slow recovery threshold," denoted as SRthresh, to represent the permitted number of times for adjustment. That is, if the system performance has not improved after we update the system for SRthresh times, it switches from "slow recovery" to "fast adjustment" phase. In other words, for the former, we linearly increase or delete m; for the latter, we exponentially adjust m.

Algorithm 1 shows the pseudocode of our proposed adjustment model. At current day t, given system performance $\kappa(t)$ and by using current training set $\underline{T}_m(t)$ and reference performance κ_0, we use a user's feedback $u(t+1)$ to perform the adjustment.

Algorithm 1 Slow Recovery and Fast Adjustment

```
Require: SRthresh
1: $\epsilon(t) = \kappa_0 - \kappa(t)$;
2: $i = 1$;
3: if $\epsilon(t) > e_{thr}$ then
4:   while $\epsilon(t) > e_{thr}$ do
5:     if $i <$ SRthresh then
6:       $\hat{m} \leftarrow m \pm 1$;
7:     else
8:       $\hat{m} \leftarrow m \times 2^{\pm(i-\text{SRthresh})}$
9:     end if
10:    Make recommendations based on $\underline{T}_{\hat{m}}(t)$.
11:    Calculate error signal $\epsilon_1(t)$ and $\epsilon_2(t)$.
12:    if $\epsilon_1(t) < e_{thr}$ or $\epsilon_2(t) < e_{thr}$ then
13:      return $\hat{m}$;
14:    end if
15:    $i \leftarrow i + 1$;
16:    Choose $\hat{m}$ with $\min(\epsilon_1(t), \epsilon_2(t))$;
17:  end while
18: else
19:   return No adjustment should be performed.
20: end if
```

E. Recommendation List Controller

As mentioned earlier, the ratio of new and old items can be quite different for each user and can largely reflect a user's viewing preferences/behavior. Therefore, it is important to observe the trend of this parameter in a user profile before making recommendations, and adjust the composition of the final recommendation list according to this value. Here, we call it as the "revisiting ratio," denoted as $\lambda(t)$. Specifically, new items indicate those items that a user has viewed before, while old ones are the others. The value is calculated by the ratio of new and old items. $\lambda(t) \in (0, 1)$ indicates that a particular user prefers old items, while $\lambda(t) > 1$ indicates that he/she prefers new ones more. However, user behaviors are dynamic, which makes it very challenging to predict exactly what they will watch next, and the only possible solution is from long-term observations. Here, we propose another feedback control loop, the recommendation list controller, to monitor the dynamics of user preference and balance the number of new and old items in the final recommendation list.

Similar to the training set controller, this controller also consists of an evaluation metric of system performance, an error signal, a control function, and a control signal. Every time the system receives a user feedback, it calculates the "revisiting ratio" $\lambda(t+1)$ and compares it with $\lambda(t)$ to evaluate the current system performance. Without loss of generality, the lower value of this error signal indicates better system performance. After, we input the error signal into a proposed prediction model (the control function) to produce $\lambda(t+1)$.

The PID controller is a time-dependent method, to consider the system performance within a period rather than the last result. Its basic idea is to use system error and its accumulation in the time domain to make the control signal. Since accurate prediction on user behaviors in our system also needs long-term observations, it is necessary to consider user profiles within a period of time when we predict $\lambda(t+1)$. Thus, we design a prediction method based on the PID control theory. Formally, we have the prediction function as

$$\lambda(t+1) = P \times \varsigma(t) + I \times \sum_{i=0}^{t} \varsigma(i)\Delta(i) + D \times \varsigma(t) - \varsigma(t-1) \tag{9}$$

where P, I, D are three parameters in this model, representing the proportional gain, the integral gain, and the derivative gain. Similar to the training set controller, we set a limit of the error signal. Every time $\varsigma(t)$ exceeds the threshold, denoted as ς_0, we believe that a user's preference has changed, and thus, an update to the current prediction model should be performed. Therefore, we use OLS to recalculate the parameters in this model (i.e., P, I, D) based on the latest user profiles.

In statistics, OLS is a method for estimating the unknown parameters in a linear regression model. This method minimizes the sum of squared vertical distances between the observed responses in the data set, and the responses predicted by the linear approximation. This technique can be applied to single or multiple explanatory variables and also categorical variables that have been appropriately coded. The basic form of the OLS regression model is

$$\hat{Y} = \hat{\beta_0} + \hat{\beta_1}X_1 + \hat{\beta_2}X_2 + \cdots + \hat{\beta_k}X_k \tag{10}$$

where $\hat{Y}$ is the response variable, k is the number of prediction variables, X_k is the continuous explanatory variable, and $\hat{\beta_k}$ is the regression coefficient of X_k. Our goal is to obtain a set of X_k with minimum $Y - \hat{Y}(Y$ that represents the real value of the response variable or minimize the sum of squared residuals. We first give these parameters tentative values, revise them slightly to see if it can be improved, and finally repeat this revision until the error is minimum.

The detailed procedure of our recommendation list controller is described as follows.

Step 1: System performance evaluation: We take $\varphi(t+1)$ as the latest user preference, to evaluate the current system performance. Every time we receive it, we compare it with $\lambda(t)$ and calculate the error signal $\varsigma(t)$ as

$$\varsigma(t) = \frac{\varphi(t+1)}{\varphi(t+1)+1} - \frac{\lambda(t)}{\lambda(t)+1}. \tag{11}$$

Step 2: Compare the error signal with the reference value: If $\varsigma(t) < \varsigma_0$, the current $\lambda(t)$ can still reflect a user's preference, and no adjustment is needed; go to **Step 4**. Otherwise, we consider that this user's behavior has largely changed, and thus, an update to the model should be performed.

Step 3: Update the prediction model: Three parameters are considered in this model, namely, P, I and D, representing the proportional gain, the integral gain, and the derivative gain, respectively. We adopt the OLS regression method to estimate the aforementioned three parameters. During this process, the latest user profile (i.e., data from the last update until now) is accumulated to be the training set of regression.

Step 4: Generate a new control signal: We adopt the proposed prediction model to generate the control signal $\lambda(t+1)$.

TABLE II
PARAMETER SETTINGS USED IN EXPERIMENTS

Parameter	a	b	c	d	k_0	ϵ
Value	0.006	0.27	0.50	0.22	0.30	0.70
Parameter	SRthresh	P	I	D	ς_0	η
Value	3	0.62	0.54	0.02	0.15	0.3

VI. Performance Evaluation

In this section, we extensively evaluate its correctness and effectiveness on a real data set. We first introduce our experimental environment and then show the results. Since there is no direct information showing a user's interests to channels (e.g., through user ratings) in our data set, without loss of generality, we use the number of viewing times to indicate his/her preference. That is, the higher the frequency that a user clicks an item, the more he/her likes it. Based on this, open source project Myrrix [37] is selected to build explicit ratings of user and items to recommend the new items.

A. Simulation Environment

We use Apache Hadoop, an open-source project to develop the reliable, scalable, and distributed computing platform in the cloud, as the simulation environment. Considering the scale of our used data set, we implement a Hadoop cluster with 11 nodes, one as the master node and the others as data nodes. These nodes are all virtual machines supported by VMware with Intel Core i-5 CPU, 20G disk and all running Ubuntu 12.04. All virtual machines are connected with a 10-Mb/s switch as a local network. Table II shows the parameter settings used in our experiments.

B. System Level Performance

In this section, we conduct a set of simulations to explicitly evaluate the performance of our feedback control-based recommender system.

First, we investigate the stability of our proposed two feedback controllers. As mentioned earlier, system dynamics is the main reason for resource cost and loss of accuracy. By self-monitoring and self-adjusting the system, our proposed feedback controllers can successfully reduce the performance loss. We randomly choose 100 users and make recommendations for ten consecutive days. Fig. 7 shows the attained average accuracy of all users and its error bar, with respect to different sizes of training set. It is clear that the benchmark approach (i.e., without controllers) cannot sustain high accuracy (abrupt changes are observed) when we change the size of the training set. However, our proposed approach improves the stability of attained accuracy. This is due to the fact that our proposed training set controller is able to continuously monitor the system status and performs updates on $\underline{T}_m(t)$, including increasing or deleting m and adjusting $\underline{T}_m(t)$ to an appropriate size. Therefore, our system can keep providing a high degree of accuracy.

Fig. 8 shows the impact of the number of recommendations on the achieved accuracy. First, we randomly select three users, recommend them the different but fixed number of channels

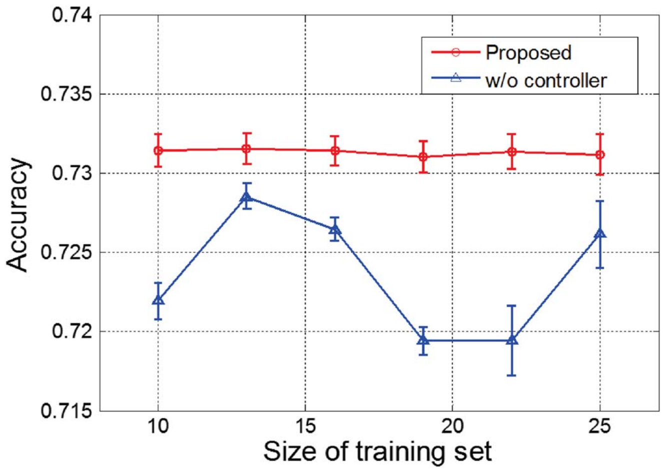

Fig. 7. Attained average accuracy of all users and its error bar, with respect to different sizes of training set.

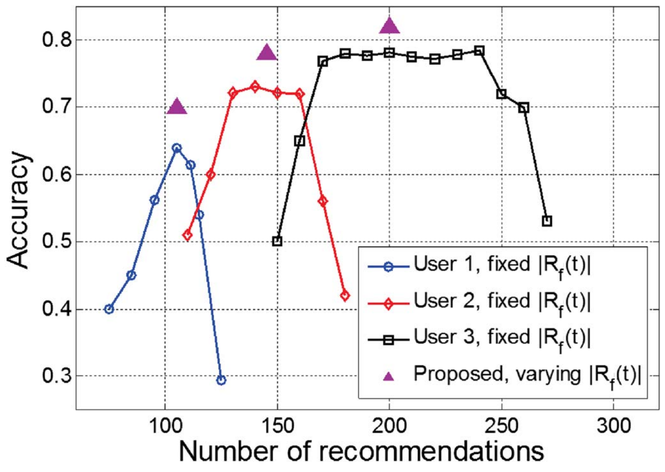

Fig. 8. Impact of the number of recommendations on the achieved accuracy.

for 30 days, and show the curve of achieved accuracy. Then, we compare with our proposal, i.e., given the predicted time-varying $|R_f(t)|$ for 30 days, we compute the average number of recommended channels and the corresponding achieved average accuracy (see the purple triangle in the figure). For fixed $|R_f(t)|$, we observe that the highest accuracy for user 1a appears when $|R_f(t)| = 110$. However, for user 2, this value expands to a range. This indicates that his/her behavior has more dynamics. User 3's behavior also has some randomness since he/she does not have a fixed watching frequency. In comparison, in our proposed solution, $|R_f(t)|$ is predicted on the daily basis, and thus, it varies according to the user behavior. However, fixing the number of recommendations (i.e., $|R_f(t)|$ does not vary) can only make accurate predictions when a user happens to always watch the same number of channels, and this is why our proposal achieves certain performance gain.

Next, we focus on the effectiveness of our proposed adjustment model, i.e., "slow recovery" and "fast adjustment." In this experiment, we use watching logs for 20 days as the initial training set. Each day, we make recommendations with new training sets and compute the error signal $\epsilon(t)$ until it is less than

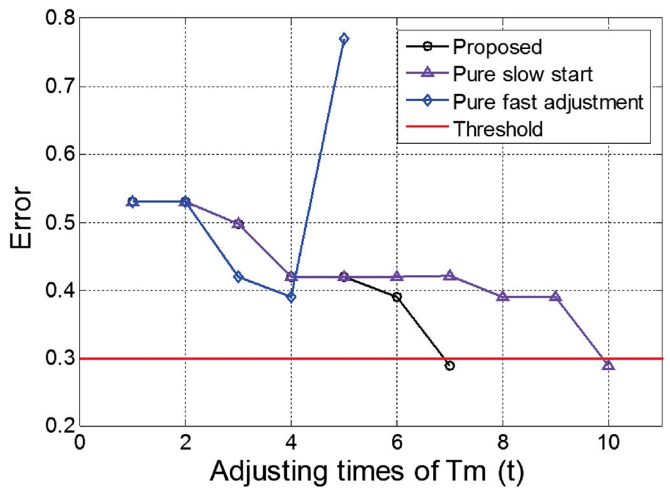

Fig. 9. Simulation results for the effectiveness of our proposed adjustment model.

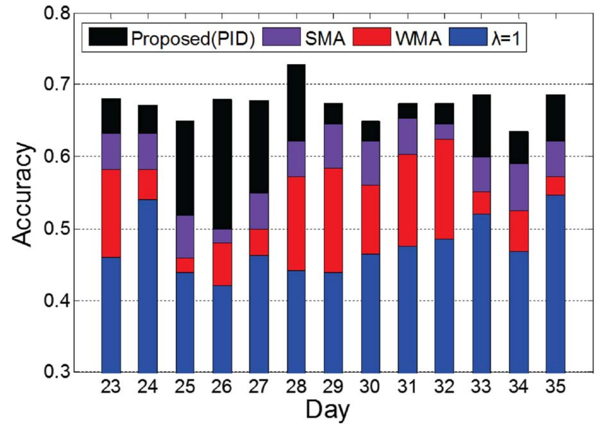

Fig. 10. Impact of "revisiting ratio" on different measurements.

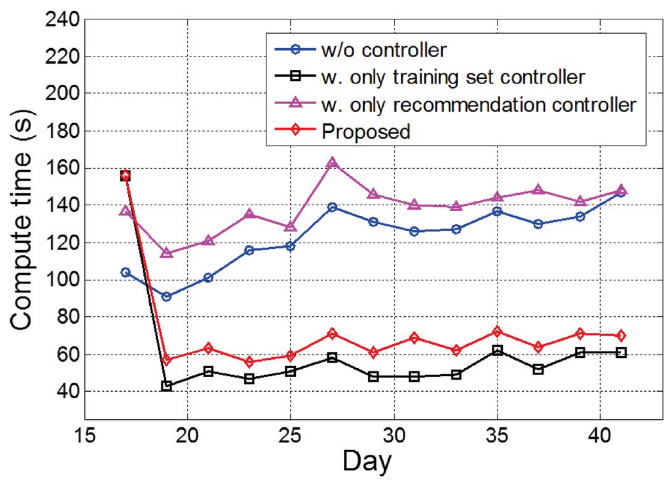

Fig. 11. Simulation results for achieved accuracy over time.

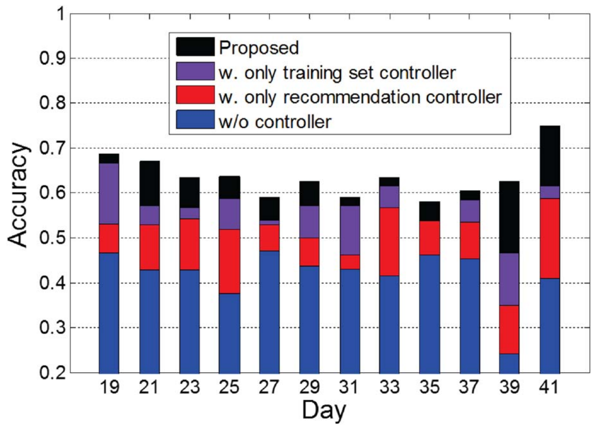

Fig. 12. Simulation results for the consumed compute time over time.

the threshold k_0 $(= 0.3)$ or does not have enough remaining training data to add. As shown in Fig. 9, our proposal is the first to reach the predetermined threshold only after seven adjustments. The "slow recovery only" strategy (i.e., SRthresh $= \infty$) achieves the same goal with three more adjustments than our proposal. Although the "fast adjustment only" strategy (i.e., SRthresh $= 1$) performs well at the beginning, it has the worst performance at the end because it deletes data too fast and this causes the data deficiency problem. Therefore, our proposal can make a balance between adjustment speed and performance, and in turn, more computational resources and update time can be saved.

Finally, we investigate the impact of "revisiting ratio" $\lambda(t)$ on different measurements, as shown in Fig. 10. We compare our proposed scheme (i.e., to adjust $\lambda(t)$ by using the PID controller), with using simple moving average (referred to as "SMA") and using weighted moving average (referred to as "WMA"), with the same number of new items and old items (referred to as $\lambda = 1$). As shown in the figure, our proposal is better than all other three approaches since the enforced recommendation controller successfully monitors the change of user-friendliness and self-adjusts the preference model.

C. Compute Time and Accuracy Over Time

One of the benefit of our proposal is the capability to handle the tradeoff between computational cost and achieved accuracy. This can help service providers have a reasonable estimate for resources. Fig. 11 depicts the computational efforts (measured in compute time) for the benchmark approach (i.e., without the controller), with only the training set controller, with only the recommendation controller, and with our proposal. Fig. 12 further demonstrates the accuracy over time. It is obvious that the benchmark approach needs almost 130 s to make recommendations, where it consumes time to build up a model with a larger scale of training set. However, when we adopt our proposed training set controller to dynamically adjust the size of the training set, the computational effort can be reduced by up to 57% (see day 27). However, when we evaluate the accuracy of these four systems as shown in Fig. 12, we observe that our solution can raise the accuracy by 10%, compared with the no-controller solution. Once the controllers start to update the system, the performance becomes stable. The reasons are due to the optimal training set with the adjustment of the training set controller, and the successful following of user preferences. Since our recommendation list controller keeps observing the trend of a user's preferences, our recommendations are always based on his/her most recent interests. Therefore, our proposal achieves the best performance over time.

VII. Concluding Remarks

In this paper, we proposed to use feedback control theory to design and implement a personalized multimedia recommendation system for cloud-integrated CPSs, where a typical application is to recommend videos on smart devices in a community. We first investigated the patterns of user behaviors from a real data set and designed a personalized recommender system to recommend both the historical and new items. Then, a new concept of revisiting ratio is introduced to represent the percentage of new recommended items and old ones, and a newly proposed preference model is adopted to make new recommendations. Furthermore, we designed two feedback controllers, the training set controller and recommendation list controller, to self-monitor the status of time-varying system and decide when and how to make an operational parameter update. Extensive experimental results have shown that our system not only reduced the cost of resources but also effectively increased the recommendation accuracy.

Future work spans different directions. First, although our proposal achieves high accuracy, there is an ongoing discussion in relation to how to find the best order of recommendations. Leveraging learning-to-rank method will lead to more effective recommender system. Second, rather than building a generalized model for all users, a user-dependent model could be investigated to autonomously adjust the parameters according to diverse user behaviors. Finally, we plan to design a new evaluation system since existing ones only consider the occurrences of recommended items but ignore the viewing times. If users view items made by our system for many times, the performance of our system can be considered as good. Therefore, viewing times of recommendations will be added into the new evaluation system. A lot of work can follow.

References

[1] F. Xia, N. Asabere, A. Ahmed, J. Li, and X. Kong, "Mobile multimedia recommendation in smart communities: A survey," *IEEE Access*, vol. 1, pp. 606–624, 2013.

[2] J. Wan, D. Zhang, S. Zhao, L. T. Yang, and J. Lloret, "Context-aware vehicular cyber-physical systems with cloud support: Architecture, challenges, and solutions," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 106–113, Aug. 2014.

[3] S. Gao *et al.*, "A cross-domain recommendation model for cyber-physical systems," *IEEE Trans. Emerging Topics Comput.*, vol. 1, no. 2, pp. 384–393, Dec. 2013.

[4] J. Wan *et al.*, "VCMIA: A novel architecture for integrating vehicular cyber-physical systems and mobile cloud computing," *Springer Mobile Netw. Appl.*, vol. 19, no. 2, pp. 153–160, Apr. 2014.

[5] "Amazon," Amazon, Seattle, WA, USA, 2013. [Online]. Available: http://www.amazon.com/

[6] "Netflix," Netflix, Los Gatos, CA, USA, 2013. [Online]. Available: https://www.netflix.com/global/

[7] "Last.fm," Last.fm, London, U.K., 2013. [Online]. Available: http://www.last.fm/

[8] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. Int. J. 14th Conf. Uncertainty Artif. Intell.*, 1998, pp. 43–52.

[9] R. Baheti and H. Gill, "Cyber-physical systems," in *Proc. Impact Control Technol.*, 2011, pp. 161–166.

[10] J. Wan *et al.*, "Enabling cyber-physical systems with machine-to-machine technologies," *Indersci. Ad Hoc Ubiquitous Comput.*, vol. 13, no. 3/4, pp. 187–196, Jul. 2013.

[11] X. Yu, A. Pan, L.-A. Tang, Z. Li, and J. Han, "Geo-friends recommendation in GPS-based cyber-physical social network," in *Proc. IEEE Int. Conf. ASONAM*, 2011, pp. 361–368.

[12] H. Jafarkarimi, A. T. H. Sim, and R. Saadatdoost, "A naive recommendation model for large databases," *Int. J. Inf. Educ. Technol.*, vol. 2, no. 2, pp. 216–219, Jun. 2012.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-A case study," Def. Tech. Inf. Center (DTIC) Doc., Ft. Belvoir, VA, USA, Tech. Rep., 2000.

[14] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[15] J. Beel, M. Genzmehr, S. Langer, A. Nürnberger, and B. Gipp, "A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation," in *Proc. ACM RepSys*, 2013, pp. 7–14.

[16] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proc. ACM Conf. Digit. Lib.*, 2000, pp. 195–204.

[17] F. Ricci, L. Rokach, and B. Shapira, *Introduction to Recommender Systems Handbook*. New York, NY, USA: Springer-Verlag, 2011.

[18] Y. Mo, J. Chen, X. Xie, C. Luo, and L. Yang, "Cloud-based mobile multimedia recommendation system with user behavior information," *IEEE Syst. J.*, vol. 8, no. 1, pp. 184–193, Mar. 2014.

[19] M. Talabeigi, R. Forsati, and M. R. Meybodi, "A dynamic web recommender system based on cellular learning automata," in *Proc. IEEE ICCET*, 2010, vol. 7, pp. V7-755–V7-761.

[20] T. Jambor, J. Wang, and N. Lathia, "Using control theory for stable and efficient recommender systems," in *Proc. ACM Int. Conf. WWW*, 2012, pp. 11–20.

[21] N. Lathia, S. Hailes, L. Capra, and X. Amatriain, "Temporal diversity in recommender systems," in *Proc. ACM SIGIR*, 2010, pp. 210–217.

[22] R. Burke, "Evaluating the dynamic properties of recommendation algorithms," in *Proc. ACM RecSys*, 2010, pp. 225–228.

[23] C. Bakir and S. Albayrak, "User based and item based collaborative filtering with temporal dynamics," in *Proc. IEEE SIU*, 2014, pp. 252–255.

[24] N. Taghipour, A. Kardan, and S. S. Ghidary, "Usage-based web recommendations: A reinforcement learning approach," in *ACM Recommend. Syst. Conf.*, 2007, pp. 113–120.

[25] K. Ogata, *Discrete-Time Control Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995, vol. 2.

[26] S. Parekh *et al.*, "Using control theory to achieve service level objectives in performance management," *Real-Time Syst.*, vol. 23, no. 1/2, pp. 127–141, Jul.–Sep. 2002.

[27] K. Meng, Y. Wang, X. Zhang, X.-c. Xiao, *et al.*, "Control theory based rating recommendation for reputation systems," in *Proc. IEEE ICNSC*, 2006, pp. 162–167.

[28] L. Bohang, Y. Xiaoyong, L. Qingbing, and H. Shougang, "An improved method for traffic control relying on close-loop control theory," in *Proc. IEEE Inf. CAR*, 2010, vol. 3, pp. 48–50.

[29] B. Agaran, W. W. Buchanan, and M. Yurtseven, "Regulating bullwhip effect in supply chains through modern control theory," in *Proc. IEEE Portland Int. Center Manage. Eng. Technol.*, 2007, pp. 2391–2398.

[30] S. Renaud-Deputter, T. Xiong, and S. Wang, "Combining collaborative filtering and clustering for implicit recommender system," in *Proc. Int. Conf. AINA*, 2013, pp. 748–755.

[31] H.-J. Kwon and K.-S. Hong, "Collaborative iptv content recommendation method using an implicit attribute preference," in *Proc. IEEE ICCE*, 2012, pp. 570–571.

[32] F. C. Stevens, "Knowledge-based assistance for accessing large, poorly structured information spaces," Ph.D. dissertation, Dept. ACM, Univ. Colorado, Boulder, CO, USA, 1993.

[33] M. Morita and Y. Shinoda, "Information filtering based on user behavior analysis and best match text retrieval," in *Proc. ACM SIGIR*, 1994, pp. 272–281.

[34] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decision making," *IEEE Trans. Syst., Man Cybern.*, vol. 18, no. 1, pp. 183–190, Jan./Feb. 1988.

[35] H. Hashemi, N. Yazdani, A. Shakery, and M. Naeini, "Application of ensemble models in web ranking," in *Proc. IST*, Dec. 2010, pp. 726–731.

[36] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "Letor: A benchmark collection for research on learning to rank for information retrieval," *Inf. Retrieval*, vol. 13, no. 4, pp. 346–374, Aug. 2010.

[37] A. Myrrix, Welcome to Apache Myrrix, 2013. [Online]. Available: http://myrrix.com/

[38] S. Rendle and L. Schmidt-Thieme, "Online-updating regularized kernel matrix factorization models for large-scale recommender systems," in *Proc. ACM RecSys*, 2008, pp. 251–258.

[39] H.-P. Shiang and M. van der Schaar, "A quality-centric tcp-friendly congestion control for multimedia transmission," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 896–909, Jun. 2012.

Chi Harold Liu (M'10) received the B.Eng. degree from Tsinghua University, Beijing, China, and the Ph.D. degree from Imperial College, London, U.K.

He is a Full Professor at the School of Software, Beijing Institute of Technology, Beijing, China. He also has worked for IBM Research and Deutsche Telekom Laboratories before moving to academia. His current research interests include the Internet of things, big data, and mobile computing. He has published more than 60 prestigious conference and journal papers and owned more than ten European Union (EU)/U.S./China patents.

Dr. Liu is a member of Association of Computing Machinery (ACM).

Zhen Zhang received the B.Eng. degree in software engineering from the Beijing Institute of Technology, Beijing, China, in 2014, where she is currently working toward the M.Sc. degree on cloud computing and big data.

Min Chen (M'08–SM'09) received his Ph.D. degree from South China University of Technology, Guangzhou, China.

He is a Full Professor at the School of Computer Science and Engineering, Huazhong University of Science and Technology, Wuhan, China. He was an Assistant Professor with Seoul National University, Seoul, Korea. He has published more than 150 papers. He serves as an Editor or Associate Editor for Wireless Communications and Mobile Computing, 24 Communications, The Institution of Engineering and Technology (IET) Networks, KSII Transactions on Internet and Information Systems, International Journal of Sensor Networks, etc.