

An effective Web page recommender using binary data clustering

Rana Forsati¹ · Alireza Moayedikia² · Mehrnough Shamsfard³

Received: 2 February 2014 / Accepted: 6 April 2015 / Published online: 28 April 2015
© Springer Science+Business Media New York 2015

Abstract Through growth of the Web, the amount of data on the net is growing in an uncontrolled way, that makes it hard for the users to find the relevant and required information- an issue which is usually referred to as information overload. Recommender systems are among the appealing methods that can handle this problem effectively. These systems are either based on collaborative filtering and content based approaches, or rely on rating of items and the behavior of the users to generate customized recommendations. In this paper we propose an efficient Web page recommender by exploiting session data of users. To this end, we propose a novel clustering algorithm to partition the *binary* session data into a fixed number of clusters and utilize the partitioned sessions to make recommendations. The proposed binary clustering algorithm is scalable and employs a novel method to find the representative of a set of binary vectors to represent the center of clusters—that might be interesting in its own right. In addition, the proposed clustering algorithm is integrated with the *k*-means algorithm to achieve better clustering quality by combining its explorative power with fine-tuning power of the *k*-means algorithm. We have performed extensive experiments on a real dataset to demonstrate the advantages of proposed binary data clustering methods and Web page recommendation algorithm. In particular, the proposed recommender system is compared to previously published works

✉ Rana Forsati
forsati@cse.msu.edu

Alireza Moayedikia
moayedikia.alireza@gmail.com

Mehnough Shamsfard
m_shams@sbu.ac.ir

¹ Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA

² Department of Information Systems and Business Analytics, Deakin University, Burwood, Victoria, Australia

³ Natural Language Processing (NLP) Research Laboratory, Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G. C., Tehran, Iran

in terms of minimum frequency and based on the number of recommended pages to show its superiority in terms of accuracy, coverage and F-measure.

Keywords Recommender systems · Binary data clustering · k -Means · Harmony search optimization

1 Introduction

Information overload stemming from the high volume of available information on the Web has made reliance on information management systems like recommender systems (RS) inevitable. RS play a significant role in many online applications such as Amazon, iTunes and Netflix, guiding the search process and helping users to effectively find the information or products that they are looking for Adomavicius and Tuzhilin (2005). RS identify a subset of items (e.g., products, movies, books, music, news, and Web pages) that are likely to be more interesting to users based on their interests (Forsati and Meybodi 2010). Mainly RS can be divided into two main categories: *content-based* (CB) and *collaborative filtering* (CF) (Koren et al. 2009).

In CB recommendation, the recommendation process relies on explicit content and textual data about the entities in the system to make recommendations. Such data can be used to measure a criterion applicable for recommendation (e.g., degree of similarity) and recommend items similar to those a given user preferred in the past (Wang et al. 2008). Dependency on the external information such as explicit item descriptions, profile of users, and/or the appropriate features extracted from items are required in order to analyze item similarity or user preference (Koren et al. 2009) to generate recommendations.

On the other hand, CF recommendation, the most popular method adopted by contemporary RS, works based on the assumption that similar users on similar items express similar interest. The system depends on the observed rating information of users in the past (Yang et al. 2011) to build a model out of the rated information. In contrast to CB based methods, CF based recommenders do not require to have access to external information. The essence of CF lies in analyzing the neighborhood information of past user-item interactions to generate recommendations based on the similarity of the users preferences and behavior (Yang et al. 2011). The CF algorithms are mainly divided further into two *memory-based* methods (Chen et al. 2009) and *model-based* methods (Zhang et al. 2006).

Memory-based CF utilizes the past information of the users and their interactions with the items/pages, to extract patterns for making recommendations to the future active users based on a criterion, like similarity (Billsus and Pazzani 1998; Breese et al. 1998; Wei et al. 2005). The most significant aspect of memory-based CF algorithms is the measurement of similarity (Zhou et al. 2008) which is based on ratings of items given by users through either the similarity of users (Herlocker et al. 1999) or the similarity of items (Wang et al. 2006).

A model-based CF uses the patterns extracted from the users' data and predicts the unknown future behavior of the users and to make recommendations. In model-based CF the goal is to employ statistical and machine learning techniques to learn models from the data and make recommendations based on the learned model. Methods in this category include aspect models (Hofmann 2004), clustering methods (Kohrs and Meriardo 1999), Bayesian models (Zhang and Koren 2007), and matrix factorization (MF) (Salakhutdinov and Mnih 2008).

Due to efficiency of model-based algorithms in handling very huge datasets, clustering based methods have become one of the most popular models among the model-based methods. They select neighbors who have similar preferences for certain items and group them into clusters (Sarwar et al. 2000a; Sarwar et al. 2000b). These techniques then predict customers preferences about the items through the results of evaluation on the same items for the cluster as a whole.

In Web browsing systems users' activities are traceable while they are surfing the Web (Talabeigi et al. 2010b). These traces can be modeled in the form of sessions, that each user has seen a number of desired pages. This visiting pattern can be considered as a rich source of knowledge for some systems to interpret and extract the behavioral patterns of the users. For example, one method for using the visiting patterns of users for knowledge extraction is *session clustering*, in which the users' sessions are clustered based on a criterion, depending on the algorithm. It then utilizes the best cluster with respect to a given test user in selecting the neighbors with higher similarities and then recommendations are made to them based on the cluster that they are settled in.

The quality of clustering method that is employed to partition the session data into a fixed number of most similar partitions, has a significant impact on the performance of RS. This issue becomes more challenging when one deals with *binary representation* for users' sessions as in Web page recommendation systems where each Web page is associated with a binary $\{0, 1\}$ value in the session, i.e., one if the page is visited by user and zero otherwise. The overarching goal of this paper is to devise an efficient clustering method for binary session data and exploit it to propose an effective Web page RS. The ultimate goal is to develop a RS that is scalable to large datasets and capable of handling sparse session data.

To cluster binary data, the harmony search (HS) optimization method is used to perform clustering, through modeling the algorithm as a combinatorial optimization problem that aims to find an optimal part of the solution space that best satisfies an optimization function, and clusters the data. The HS algorithm is a search heuristic based on the improvisation process of jazz musicians. It was inspired by the observation that the aim of music is to search for a perfect state of harmony. In jazz music the different musicians try to adjust their pitches, such that the overall harmonies are optimized due to aesthetic objectives. Starting with some harmonies, they attempt to achieve better harmonies by improvisation. This is analogous to optimizing a given objective function instead of harmonies. Here the musicians are identified with the decision variables and the harmonies correspond to solutions. The HS was first developed by Geem et al. (2001), though it is a relatively new metaheuristic algorithm, its effectiveness and advantages have been demonstrated in various applications (Forsati et al. 2008; Mahdavi et al. 2008; Forsati et al. 2013; Mirkhani et al. 2013; Forsati and Mahdavi 2010; Forsati et al. 2013).

The HS algorithm initializes the pool of solutions that is called harmony memory with randomly generated solutions. The number of solutions is a fixed and usually depends on the number of decision variables. Then iteratively a new solution is created as follows. Each decision variable is generated either on memory consideration and a possible additional modification, or on random selection. The parameters that are used in the generation process of a new solution are called harmony memory considering rate (HMCR) and pitch adjusting rate (PAR). Each decision variable is set to the value of the corresponding variable of one of the solutions in the harmony memory with a probability of HMCR, and an additional modification of this value is performed with a probability of PAR. Otherwise, with a probability of $(1 - \text{HMCR})$, the decision variable is set to a random value. After a new solution has been created, it is evaluated and compared to the worst solution in the

memory. If its objective value is better than that of the worst solution, it replaces the worst solution in the memory. This process is repeated, until a termination criterion is fulfilled. A remarkable strength of the HS algorithm hinges on its capability in achieving a good trade-off between exploration and exploitation that makes it more appropriate for optimization problems with a complex solution space such as binary data clustering.

The problem associated with most of the optimization algorithms is their entrapment in local optima. We integrated HS with a local search procedure, in our case k -means method, to avoid the solution from trapping in local optima through effective exploration of the search space. As a result, our method exhibits both high accuracy and computational efficiency as shown in the experiments we have conducted on DePaul University CTI log file dataset.

1.1 Summary of contributions

This paper makes the following contributions:

- An efficient clustering algorithm to partition binary data points into a predefined number of clusters and a novel method to find the representative of a set of binary vectors to represent the center of clusters in binary clustering methods.
- Hybrid clustering algorithms by exploiting the explorative power of HS optimization method and fine-tuning power of localized k -means clustering method.
- An effective Web page RS by applying the proposed binary data clustering method to the session data extracted from users' log file.
- An extensive set of experiments on real datasets to investigate the performance of proposed clustering and recommendation algorithms. In addition we demonstrate the merits and advantages of proposed recommended system in handling large and sparse datasets.

1.2 Outline

The remainder of the paper is organized as follows. In Sect. 2 a comprehensive survey of the literatures on session data clustering and RS is given. The general architecture of the proposed RS with explanation of its main modules are given in Sect. 3. The proposed clustering algorithm is discussed in Sect. 4. The time complexity of the proposed algorithms are analyzed in Sect. 5. Section 6 includes the extensive experiments we have performed to evaluate the proposed session clustering and recommendation algorithms. Finally, Sect. 7 concludes the paper and discusses few directions as future work.

2 Literature review

Clustering Web sessions is a part of Web usage mining which is the application of data mining techniques to discover usage patterns from Web data typically collected by Web servers in large logs. Session clustering algorithms can be adopted from different perspectives including, representation schema and clustering techniques. Sessions of the users can be represented in vector and non-vector models. Vector models by binary representation for Web sessions have been used by many authors. Using non-vector model in order to cluster Web users is seen in several Web session clusterings.

The major application areas for Web session clustering fall into RS. Since the main goal of this work is to propose a new session clustering algorithm and then utilize it in a RS, hence we provide the literature review of Web session clustering and then RS that use session clustering as a building block for recommendation.

2.1 Related work on clustering Web sessions

User clustering algorithms are based on two representation models of vector and non-vector based schemes. In vector model the log data of users is represented in the form of binary which considers 0 and 1 as exclusion and inclusion of data, while in the other type the data is represented in a non-binary form. Table 1 summarizes the session clustering algorithms discussed below.

Fu et al. (1999, 2000) applied the BIRCH clustering algorithm (Zhang et al. 1996) to generalized session space. Access patterns of the Web users are extracted from Web server’s log files, and then organized into sessions which represent episodes of interaction between Web users and the Web server. Using attributed oriented induction, the sessions are then generalized according to the page hierarchy which organizes pages according to their generalities. The generalized sessions are finally clustered using a hierarchical clustering method. However, problem of BIRCH is its requirement to setting of a similarity threshold and sensitivity to the order of data input.

The COWES system as a novel Web user clustering method is proposed by Chen et al. (2009). This algorithm clusters users based on the evolution of the Web usage data. The motivation was that most of the existing approaches cluster Web users based on the snapshots of usage data, while the usage data are evolutionary in the nature. Consequently, the usefulness of the knowledge discovered by existing Web user clustering approaches might be limited. This limitation is addressed by clustering users according to the evolution of usage data. For a set of users and the relevant historical data, the COWES investigates how their usage data change over time and mine evolutionary patterns from each users usage history. The discovered patterns capture the characteristics of changes to a Web users information needs. The Web users are later clustered by analyzing common and

Table 1 The classification of related works based on representation schema and clustering method utilized in the recommendation process

Research study	Representation scheme		Clustering technique
	Vector based	Non-vector based	
Fu et al. (1999, 2000)		✓	BIRCH
Chen et al. (2009)		✓	Evolutionary
Ypma et al. (2002)		✓	Hidden Markov model (HMM)
Kumar et al. (2007)	✓	✓	Rough set theory-based algorithms
Duraiswamy et al. (2008)		✓	Agglomerative technique
Madhuri et al. (2011)		✓	Variable-length Markov chains
Park et al. (2008)		✓	Fuzzy ART-enhanced <i>k</i> -means algorithm
Kuo et al. (2005)		✓	ART2 neural network and genetic <i>k</i> -means
Kim and Seog (2007)		✓	Self-organizing map based clustering
Jalali et al. (2008)	✓		Graph partitioning
Wan et al. (2012)		✓	Random indexing

similar evolutionary patterns. The generated Web user clusters will provide valuable knowledge for personalization applications.

A mixture of hidden Markov models for modeling clickstream of Web surfers is proposed by Ypma et al. (2002). Hence, this data is used for Web page categorization without the need for manual categorization. To train the mixture of hidden Markov model an EM algorithm is used. This training shows that additional static user data can be incorporated easily to possibly enhance the labelling of users.

Kumar et al. (2007), proposed a general sequence-based Web usage mining methodology through utilizing new sequence representation schemes in association with Markov models. Their algorithm uses a new similarity measure namely as S^3M that is applicable to Web data. The S^3M measure considers both the order of occurrence of items as well as the content of the set. Also the paper addresses the use of sequential information in clustering Web users. Their approach considers the concept of constrained-similarity upper approximation that represents a new rough agglomerative clustering approach for sequential data. The proposed approach enables merging of two or more clusters at each iteration and hence facilitates fast hierarchical clustering. This new methodology was tested by examining cluster recovery rate as a clustering performance measure with different sequence representations and different clustering methods. The algorithm also shows that identifying Web user clusters through sequence-based clustering methods helps recover the Web user clusters correctly. As more sequential information is given, more accurate cluster recovery becomes possible.

Duraiswamy et al. (2008) presented another session clustering algorithm that focuses on clustering the session details obtained from log files that are obtained after the preprocessing of log file. Sequence alignment is used to find the similarity between every pair of sessions. This similarity is measured through dynamic programming. Similarity matrix is clustered using agglomerative hierarchical technique in order to group the same similarity session.

Madhuri et al. (2011) developed a new sequence representation scheme in association with variable length Markov models through general sequence-based Web usage mining methodology. The algorithm focuses more on the use of sequential information in clustering Web users. The new methodology was tested by examining cluster sum of squared error (SSE) as a clustering performance measure.

Park et al. (2008) utilized a new sequence representation scheme in association with Markov models to develop a sequence-based clustering method. The new sequence representation calculates vector-based distances (dissimilarities) between Web user sessions and thus can be used as the input of various clustering algorithms.

Kuo et al. (2005) challenged the determination of number of clusters and proposed an algorithm to address this issue. The authors proposed a two-stage method, which first uses the adaptive resonance theory 2 (ART2) to determine the number of clusters and an initial solution, then using genetic k -means algorithm (GKA) to find the final solution. Sometimes, the self-organizing feature map with two-dimensional output topology has great difficulty determining the number of clusters by observing the map. However, ART2 can actually determine the number of clusters according to the number of output nodes. Through Monte Carlo simulation and a real case problem, the proposed two-stage clustering analysis method, ART2 + GKA, has been shown to provide high performance. The p -value of Scheffes multiple comparison test shows that the two cluster methods, ART2 + k and ART2+GKA, do not differ significantly, but the average of within cluster variance of ART2 + GKA is less than that of ART2 + k -means. This is resulted since ART2 + GKA has the characteristics of both a genetic algorithm and k -means.

Kim and Seog (2007) proposed a novel algorithm which incorporates sequential navigation order into a clustering process. The visitors interest are calculated by weighting the importance of Web pages based on their visited orders and then forming an order-dependent representation. Also experiments indicated that the clustering results are reflecting not only how many same pages visitors visit but also how close navigation sequences of visitors are. Also the authors introduced a simple and effective method to summarize multi-dimensional clustering outputs. In particular, without resorting to another clustering algorithm, the proposed visualization system makes it possible for users to visually identify similar groups of navigation patterns. It also graphically shows that clustering outputs from two representation methods are significantly different. Furthermore state-transition diagrams are used to visualize navigation patterns of visitors for datasets based on either a boolean or an order-dependent representation. The estimated probabilities of transitions between states can be used to dynamically recommend a page for visitors to visit next.

Jalali et al. (2008) presented a study of the Web based user navigation patterns mining and proposed a novel algorithm for clustering of user navigation patterns. The algorithm is based on the graph partitioning for modeling user navigation patterns. For the clustering of user navigation patterns an undirected graph based on connectivity between each pair of Web pages was created and a new methodology for assigning weights to edges in such a graph was proposed.

Wan et al. (2012) proposed a Web user clustering approach to prefetch Web pages for grouped users based on random indexing. Segments split by '/' in the URLs are used as the unit of analysis in their study. The random indexing model is constructed to uncover the latent relationships among segments of different users and extract individual user access patterns from the Web log files. Furthermore, random indexing was modified with statistical weighting functions for detecting groups of Web users. Common user profiles can be created after clustering single-user navigational patterns.

2.2 Related work on Web page recommender systems

Cluster analysis on session data is applicable in many areas including RS. Web page recommenders that consider Web usage mining techniques can treat the recommendation process as an optimization, graph-based or machine learning algorithm. Table 2 outlines the reviewed recommender methods with their category.

The SUGGEST recommendation algorithm is proposed by Baraglia and Silvestri (2007). Their algorithm has a two-level architecture composed by an off-line creation of historical knowledge and an online engine that understands user behaviors. The system work in the way that when the requests arrive at the system, it will incrementally update a graph representation of the Web site based on the active user sessions. A graph partitioning algorithm is used to classify the active sessions using a graph partitioning algorithm. However, this architecture has its own limitations. First is the memory consumption rate which requires storage of Web server pages to be quadratic in the number of pages, which will severely affect large Web sites that are made up from millions of pages. Second, it suffers from management of Web sites that are made up of dynamic pages.

Jalali et al. (2010) proposed a Web usage mining architecture called WebPUM which is based on their previous proposal (Jalali et al. 2008). The authors' novel approach is to classify user navigation pattern for online prediction of user future intentions through mining Web server logs. To model user navigation patterns a graph partitioning algorithm was used, in which an undirected graph based on connectivity between each pair of the Web pages in order to mine user navigation patterns. Then this procedure assigns weights

Table 2 The classification of related recommender algorithms

Research study	Recommender category
Baraglia and Silvestri (2007)	Graph based recommender
Eirinaki and Vazirgiannis (2007)	
Jalali et al. (2010)	
Jalali et al. (2009)	Collaborative filtering
Shinde et al. (2008)	
Sarwar et al. (2002)	
Ntoutsis et al. (2012)	
Kyung-Yong et al. (2002)	Learning based recommender
Forsati and Meybodi (2010)	
Forsati et al. (2013)	
Talabeigi et al. (2010b)	
Talabeigi et al. (2010a)	
Mehr et al. (2011)	Machine learning and graph partitioning based recommender
AlMurtadha et al. IPACT (2011)	

to edges of the undirected graph. Also to predict the users future movements and classify current user activities the longest common subsequence (LCS) algorithm was applied.

Wang et al. (2011) investigated the problem of mining Web navigation patterns. The authors introduce the concept of throughout-surfing patterns, which are effective to predict Web site visitors surfing paths and destinations. Secondly, authors propose the path traversal graph and graph traverse algorithm to increase the efficiency of mining throughout-surfing patterns. Throughout-surfing patterns are more effective for content management and they are applicable to providing surfing paths recommendation and personalized configuration of dynamic Web sites. In addition, a path traversal graph structure is suitable for incremental mining of sequential patterns.

Shinde et al. (2008) proposed architecture to classify user navigation pattern and online recommendation to users for predication of future request by mining of Web server logs. This architecture will be used in the Web usage mining system named as ORWUMS. In their architecture, a recommendation engine that works in online phase predicts next user request by interacting user profile and classification module. After classification part, accuracy of classification will be evaluated by evaluation part.

Also Sarwar et al. (2002) presented and experimentally evaluated a new approach in improving the scalability of RS by using clustering techniques. The authors suggest that clustering based neighborhood provides comparable prediction quality as the basic CF approach and at the same time significantly improves the online performance.

Ntoutsis et al. (2012), presented gRecs, as a system for group recommendations that follows a collaborative strategy, with the aim of enhancing recommendations with the notion of support to model the confidence of the recommendations. Moreover, the algorithm partitions users into clusters of similar ones, to make recommendations for users according to the preferences of their cluster members without extensively searching for similar users in the whole user base.

Kyung-Yong et al. (2002) proposed a new algorithm for prediction using associative user clustering and Bayesian estimated value to complement the problems of the current CF system. The representative attribute neighborhood is for an active user to select the nearest neighbors who have similar preference through extracting the representative attributes that

most affects the preference. Associative user behavior pattern is clustered according to the genre through association rule hyper-graph partitioning algorithm, and new users are classified into one of these genres by naive bayes classifier. Besides, to get the similarity between users belonged to the classified genre and new users, the algorithm permits the different estimated values to items which users evaluated through naive bayes learning.

Talabeigi et al. (2010b) proposed another dynamic Web page RS based on asynchronous cellular learning automata (ACLA) which continuously interacts with the users and learns from his behavior. The system uses all factors which have influence on the quality of recommendation and might help system to be able to make more successful recommendations. The proposed system uses usage data and structure of the site to learn user navigation patterns and predicting user's future requests.

Forsati and Meybodi (2010) proposed another algorithm for Web page recommender which is based on automata algorithm. The authors propose three variations. In the first variation, distributed learning automata is used to learn the behavior of previous users and recommend pages to the current users based on the learned patterns. The second variation presents a novel weighted association rule mining algorithm for Web page recommendation. Also, in the second variation a novel method is used to pure the current session window. The third variation is a hybrid algorithm based on distributed learning automata and the proposed weighted association rule mining algorithm, which solves the problem of unvisited or newly added pages. The significances of their work is low computational and memory utilization along with superiorities of hybridization idea, over the competitors.

Mehr et al. (2011) proposed a new recommendation algorithm based on a hybrid method of distributed learning automata and graph partitioning. Hyperlink graph and usage data are used to determine the similarities between Web pages. With the assumption that the hyperlink graph of a Web site are usually such that similar pages are divided into the same partition after partitioning the hyperlink graph of the Web site, partitioning the hyperlink graph of the Web site is used to help the distributed learning automata to learn similarities between Web page more accurately. The pages in a recommendation list are ranked according to their importance and similarities which are in turn computed based on proposed algorithm and PageRank and PageRate algorithm, and then users receive recommendations according to the similarities and the usage data the algorithm recommends pages that are more likely the user to visit. The significance of the algorithm is its ability in determining similarities between Web pages better than distributed learning automata algorithms. As well as Hebbian and extended Hebbian algorithms. the other significance of the work is that for a real Web site, the proposed algorithm provides better recommendations than the other learning automata based recommendation method.

AlMurtadha et al. (2011) proposed a two phase recommender algorithm. In the first phase which offline phase the algorithm is responsible for partition the filtered sessionized transactions into clusters of similar pageviews. Then, the future Web navigation profiles are created according to these clusters using PACT methodology. The input to the offline phase is the preprocessed Web server logs file and the outputs are clusters of navigation transactions and the Web navigation profiles. The second phase which is the online phase has the responsibility of matching the new user transaction (current user session) to the profile shares common interests to the user. The inputs to this step are the navigation profiles generated from the offline phase and the current user session. The output will be the recommendation set in addition to the best profile that matches the user interests.

Eirinaki and Vazirgiannis (2007) proposed a graph based RS. They introduce link analysis in the area of Web personalization. The authors believe that a page is important if many users have visited it before. Based on this assumption a novel algorithm, named

Usage-based PageRank (UPR) is proposed that assigns importance rankings (and therefore visit probabilities) to the Web sites pages. UPR is inspired from PageRank algorithm and is applied to an abstraction of the user sessions named the navigational graph (NG). Then based on UPR the authors developed l-UPR which is a recommendation algorithm based on a localized variant of UPR. This characteristic of UPR is applied to the personalized navigational sub-graph of each user for providing fast, online recommendations. Then UPR was integrated with probabilistic predictive model (h-PPM) as a robust mechanism for determining prior probabilities of page visits.

Talabeigi et al. (2010a) proposed another machine learning approach toward the problem, which they believe is more suitable to the nature of page recommendation problem and has some intrinsic advantages over previous method. For this purpose, an ACLA for learning user navigation behavior and predicting useful and interesting pages was used. The idea of using ACLA is that in proposed ACLA there are small number of cell which each of them represents a page. Each cell can hold the traversed page and recommended page because of the characteristic of CLA and use them for rewarding or penalizing process.

Jalali et al. (2009) developed a model for classifying user navigation patterns. This model considers LCS algorithm to predict the future users' behaviors. The algorithm in its predictions considers Web usage mining of the system. The utilization of LCS led to higher rates of accuracy in recommendation when the users have long patterns of activities in the particular Web sites, as the experiments indicated.

3 The proposed recommender system

In this section we describe the proposed framework for Web page recommendation. The proposed algorithm significantly differs from the existing algorithms by introducing a novel binary session clustering algorithm into the RS.

Algorithm 1. Harmony Recommender (HR) Algorithm

Input: \mathcal{S} : the set of session vectors, K : the number of clusters, w : the window size,
 $u, \mathbf{s}, \mathbf{o}$: the current user, its session vector, and page ordering sequence
 r : the size of recommendation set

Input: \mathcal{T} : the set of target pages for recommendation

Cluster the sessions \mathcal{S} and denote the result by $\mathcal{C} = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K)$

Compute $c_* = \min_{i \in \{1, 2, \dots, K\}} \langle \mathbf{s}, \mathbf{c}_i \rangle$

Set $\mathcal{R} = \{\}$

for each $\mathbf{s}' \in \mathcal{S}_{c_*}$ **do**

if $\mathbf{s}'[o_m] = 1$ **then**

$\mathcal{R} \rightarrow \mathcal{R} \cup \{\mathbf{s}'\}$

end if

end for

Set $\mathcal{T} = \{\}$

for each $\mathbf{s}' \in \mathcal{R}$ **do**

 Let \mathbf{o}' be the ordering associated with session \mathbf{s}'

 Add the page followed by o_m in \mathbf{o}' to \mathcal{T}

end for

Rank the pages in \mathcal{T} based on the number of occurrence in \mathcal{S}_{c_*}

Return top r pages in the ranked list

The detailed steps of the proposed recommender, dubbed as *Harmony Recommender* (HR), are outlined in Algorithm 1. The general framework is depicted in Fig. 1. The HR consists of two main components: *offline* component to extract information and cluster the session data into a predefined number of clusters, and *online* component to make recommendations for active users based on the clustered sessions and most similar sessions.

A Web server’s log contains log records of users’ accesses to the site. Each record represents a page request (URL) for a Web user along with other information such as data, IP address, time, and date. From such a Web server log, for each user his/her access pattern can be extracted. A user’s access pattern, dubbed *session*, consists of the pages visited by the user and the time he/she spent on each page. The details about sequence of pages accessed by the users can be gathered by preprocessing the Web server log data. Let $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$ denote the set of pages in the site. Each session \mathbf{s} is represented by a binary vector $\{0, 1\}^m$ where m is the total number of pages and i th entry of session vector indicates whether or not the i th page has been visited by user, i.e., the value 1 if the page has been visited and 0 otherwise. The set of all sessions are denoted by $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n\}$ where n is the total number of users.

The inputs to the clustering algorithm are the set of sessions \mathcal{S} and the number of predefined clusters K . Let us represent the result of a clustering algorithm by $\mathcal{C} = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K)$ where $\mathbf{c}_i \in \mathbb{R}^n, i \in \{1, 2, \dots, K\}$ represent the center of the i th cluster. Each clustering solution partitions the set of all sessions into K subsets where the sessions belong to the i th cluster are denoted by $\mathcal{S}_i, i \in \{1, 2, \dots, K\}$ and we have $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \dots \cup \mathcal{S}_K$. We denote by $n_i = |\mathcal{S}_i|$ the number of sessions that are assigned to the i th cluster. The clustering methods we have proposed in this paper are based on the HS optimization algorithm and will be discussed in Sect. 4. In particular, we introduce three different methods to generate clusters, named harmony session clustering (HSC), HKSC, and interleaved HKSC (IHKSC), all are based on the HS algorithm and k -means clustering method. A summary of notation used in the paper is shown in Table 3.

In online phase, the clustered sessions are fed to a RS for generating recommendations for active users. Let u denote the current user (also called active user) in the system we aim

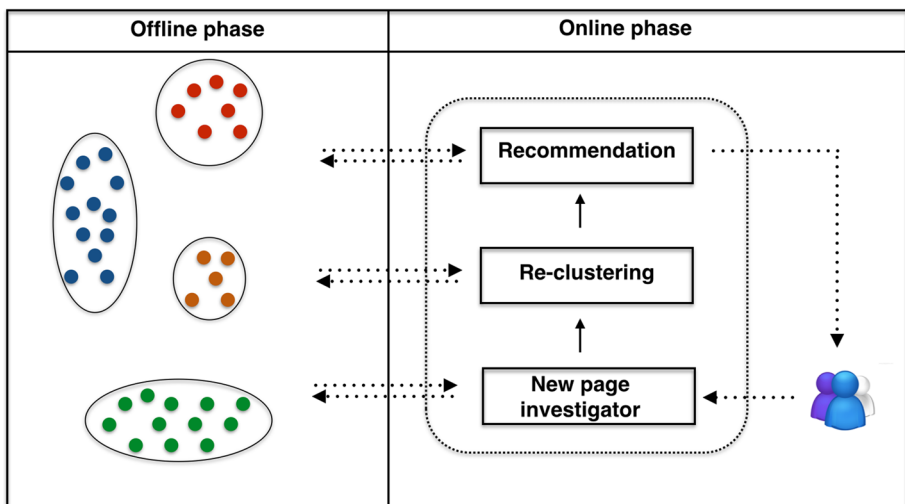


Fig. 1 The general framework of the proposed algorithm

at finding the best recommendations for his/her. We use \mathbf{s} to denote the session vector of the current user. While in the clustering users' sessions we did not take into account the order of pages visited by each user, but the recommendation process is designed to utilize the order information. Let $\mathbf{o}_i \in \{p_1, p_2, \dots, p_m\}^{m_i}$ denote the ordering of pages for the i th user where m_i is the number of pages visited by the i th user, i.e., $o_{i,1} \in \{p_1, p_2, \dots, p_m\}$ is the first page visited by i th user, $o_{i,2}$ is the second page and so on. Similarly, let $\mathbf{o} \in \{p_1, p_2, \dots, p_m\}^{n_u}$ denote the page ordering for the current user u visited so far. The first step in the recommendation process is to assign \mathbf{s} to one of the clusters in \mathcal{C} . To this end, we assign it to the cluster center that is most similar to \mathbf{s} , i.e., $c_* = \min_{i \in \{1, 2, \dots, K\}} \langle \mathbf{s}, \mathbf{c}_i \rangle$ where $\langle \cdot, \cdot \rangle$ denotes the dot product between two vectors. Then, we check all the sessions in the c_* th cluster to find a subset of these sessions $\mathcal{R} \subseteq \mathcal{S}_{c_*}$ as candidate sessions for recommendation. The selection of candidate sessions is based on the last page o_m (or few last pages) visited by the user u . In particular, for any session in \mathcal{S}_{c_*} which also has o_m visited, we include that session in \mathcal{R} .

Having computed the candidate sessions \mathcal{R} , we turn to constitute the actual recommendation set, denoted by \mathcal{T} . For every session in \mathcal{R} , the page which has been visited after the page o_m in the session is included in the set of target recommendation pages \mathcal{T} . Since, this set might include many pages, we need a mechanism to filter out the less important pages from \mathcal{T} . To this end, for each page in \mathcal{T} we count the number of sessions in \mathcal{R} the page is appeared. We consider this number as the weight of the page and utilize it to rank the pages. Finally, we choose the top-ranked pages for recommendation and the remaining pages are discarded from recommendation.

In Fig. 1, an overall process of the online recommendation is shown. In this figure we assume that the offline process has been performed successfully and the passive users are clustered. The process works for active users that are viewed as sessions containing arbitrary pages, visited by an active user. As an active user browses through the system for window size number of pages (w), then the recommendation process starts. As it is shown

Table 3 Summary of notations consistently used in the paper and their meaning

Symbol	Meaning
K	The desired number of clusters
$\mathcal{U} = \{u_1, u_2, \dots, u_n\}, n$	The set and number of users
$\mathcal{P} = \{p_1, p_2, \dots, p_m\}, m$	The set and number of pages in the system
$\mathbf{s} \in \{0, 1\}^n, \mathbf{o} \in \{p_1, p_2, \dots, p_m\}^n$	The session vector and the ordering of visited pages for a user
$\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n\}$	The set of session vectors for all users
$\mathcal{C} = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K)$	A set of centroid vectors for a clustering result where $\mathbf{c}_i \in \mathbb{R}^m$
$\mathcal{R} \subseteq \mathcal{S}$	The set of candidate sessions considered for recommendation
$\mathcal{T} \subseteq \mathcal{P}$	The set of target pages identified for recommendation
M	Harmony memory
HMS	Harmony memory size
HMCR	Harmony memory consideration rate
PAR	Pitch adjusting rate
PAR _{min} , PAR _{max}	Lower and upper bounds on pitch adjusting rate
NHV	A solution resulted from improvisation step of HS algorithm

in Fig. 1, a given active user has visited pages with ids 10, 4 and 46 in his current session. Then his new cluster is identified through measuring the similarity of the user's session vector with the cluster centroids. At this point the new cluster for the user is identified. Based on the pages visited and the other passive users reside in the newly identified cluster, the next page recommendation can be done. A key factor that requires careful consideration is the number of allowed recommendations r . In the proposed algorithm, the number of appearance of each page in the candidate sessions \mathcal{R} is utilized to rank the pages in \mathcal{T} . After ranking, the r top-ranked pages will be recommended to the active user. A more aggressive weighting schema would be to count the number of times the page appears exactly after o_m . We note that both ranking methods make the proposed algorithm less sensitive to the size of recommendation set.¹

We now turn to explaining the re-clustering module in the proposed architecture in Fig. 1. With the evolving nature of existing Web sites, RS must be able to cope with the dynamicity of Web applications. This issue might arise from two main different sources. First, many Web sites intensively use dynamic pages and therefore a RS must be able to consider these pages in recommendation process. To overcome this dynamic nature of Web sites, many recommend systems aim to leverage other types of information about Web pages to expand the set of recommendations with new pages. For example, the *content* of pages or the *structure* of Web site could provide a rich source of information to expand recommendations (Forsati and Meybodi 2010). Since the main focus of this work is only on utilizing usage information in recommendation process we will not focus on this issue.

Beyond dynamic pages, a RS must be able to react to new navigation drifts in users' behaviors and capture the appearance of new pages in session data. To handle this issue, the offline component must be periodical to have up-to-date data patterns, but the frequency of the updates is a problem that must be solved on a case-specific basis. Intuitively, the more often the clusters are being updated, the higher the accuracy achieved, but the higher the computational cost afforded, because of the extremely large volume of data being handled. However, because of the size of the number of users and session data at hand, very frequent (e.g., daily) updates are often not affordable given the available computational resources. Hence, system administrators often perform the update at fixed intervals of time (e.g., weekly, fortnightly), in an effort to balance accuracy versus cost. A question thus arises as to how to set the update frequency of the online RS, so to strike the right balance between accuracy and cost. The re-clustering module in the proposed architecture is designated to decide the right timing for clustering the sessions in order to reflect the effect of accumulated session data on the existing clusters. This module uses two parameters, namely the frequency of pages in clusters denoted by PagFreq and the re-clustering threshold. The frequency of each page in each cluster is the ratio of number of sessions the page appeared to the total number of sessions in the cluster. For each page in the active user's session if its frequency is less than the predetermined PagFreq parameter, we just update the cluster centroid. Otherwise, i.e., if the frequency is larger than equal to the PagFreq, after updating the centroid its distance to the old centroid is computed. If the distance is larger than the re-clustering threshold, the offline clustering algorithm is triggered to re-cluster all sessions.

¹ For the simplicity of exposition, in the above mentioned process we assumed that the whole session vector s of the current user is used to find the most similar cluster. Alternatively, we can use only a subset of pages from the session to assign the user to its corresponding cluster. The size of this subset is determined by a parameter which is called the *window* size. The window size is basically the number of visited pages by current user we are allowed to utilize in recommendation. In the conducted experiments we will investigate how this parameters affects the performance of the RS.

4 Clustering binary session data

In this section we introduce the binary session clustering algorithm that is used in RS. As already mentioned, the first step in the proposed recommender algorithm is to cluster session vectors into K clusters. The performance of recommendations is strongly depends on the quality of clustering algorithm in finding the hidden pattern between users' session data. Therefore, we develop novel algorithms which are more suitable to session data. We begin by introducing the HS algorithm, followed by a novel algorithm based on HS to cluster binary data. Then we introduce few hybrid clustering algorithms to combine the advantages of HS based and k -means clustering methods. We note that although HS algorithm possesses a similar structure to other existing population-based meta-heuristic solvers, but it incorporates some distinctive features that make it more appropriate for binary data clustering.

4.1 Harmony search optimization algorithm

Harmony Search is an optimization algorithm that mimics the process of improvisation in music (Lee and Geem 2005). Recent years have witnessed a flurry of research on HS to solving different optimization problems (Forsati et al. 2008; Mahdavi et al. 2008; Forsati et al. 2013; Mirkhani et al. 2013; Forsati and Mahdavi 2010; Forsati et al. 2015; Forsati and Shamsfard 2014). The main steps of the algorithm are as follows:

Algorithm 2. Basic Harmony Search

```

Initialize the algorithm parameters
Fill the harmony memory and calculate the fitness of each solution
while the stopping condition is not met do
  Generate a new harmony vector and calculate its fitness
  if the generated solution is better than the worst solution in the memory then
    Replace the generated solution with the worst one
  end if
end while

```

These steps are described in the next five subsections for general optimization problems and will be adopted to the binary data clustering in Sect. 4.2.

Initialize the parameters In Step 1, the optimization problem is specified as follows:

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

$$x_k \in [l_k, u_k] \quad k = 1, 2, \dots, n,$$

where $f(\mathbf{x})$ is the objective function and n is the number of decision variables. The lower and upper bounds for each decision variable are l_k and u_k , respectively. We note that the decision variables could be discrete or continuous or even mixed depending on the type of the problem to be solved. The objective function $f(\mathbf{x})$ captures the efficiency of a particular solution \mathbf{x} and usually referred to as *fitness* function. Additionally, we do not assume any particular structure for the objective function such as convexity which makes the optimization problem much harder.

The HS parameters are also specified in this step. These are the harmony memory size (HMS), or the number of solution vectors in the harmony memory, the probability of

memory considering (HMCR), the probability of pitch adjusting (PAR), and the number of improvisations (NI), or stopping criterion. The harmony memory, denoted by \mathbf{M} , is a memory location where all the solution vectors (sets of decision variables) are stored. The harmony memory is similar to the genetic pool in the genetic algorithms. The HMCR, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the \mathbf{M} , while $(1 - \text{HMCR})$ is the rate of randomly selecting one value from the possible range of values.

In contrast to single-point search-based algorithms in which a unique solution is generated at each iteration, the HS algorithm maintains a set of solutions in \mathbf{M} which evolve at each iteration. Therefore, HS provides an efficient and natural way for exploring the search space and obtaining an acceptable solution.

Initialize the harmony memory In this step, the harmony memory \mathbf{M} is filled with as many randomly generated solution vectors as the HMS:

$$\mathbf{M} = \left[\begin{array}{cccc|c} \mathbf{x}_1^1 & \mathbf{x}_2^1 & \dots & \mathbf{x}_{n-1}^1 & \mathbf{x}_n^1 & f(\mathbf{x}^1) \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \dots & \mathbf{x}_{n-1}^2 & \mathbf{x}_n^2 & f(\mathbf{x}^2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_1^{\text{HMS}-1} & \mathbf{x}_2^{\text{HMS}-1} & \dots & \mathbf{x}_{n-1}^{\text{HMS}-1} & \mathbf{x}_n^{\text{HMS}-1} & f(\mathbf{x}^{\text{HMS}-1}) \\ \mathbf{x}_1^{\text{HMS}} & \mathbf{x}_2^{\text{HMS}} & \dots & \mathbf{x}_{n-1}^{\text{HMS}} & \mathbf{x}_n^{\text{HMS}} & f(\mathbf{x}^{\text{HMS}}) \end{array} \right],$$

where each row of the \mathbf{M} corresponds to a single solution with its associated fitness value.

The initial harmony memory is generated from a uniform distribution in the ranges $[l_i, u_i]$, where $1 \leq i \leq n$. This is done as follows:

$$x_i^j = l_i + r \times (u_i - l_i), \quad j = 1, 2, \dots, \text{HMS}.$$

where $r \sim U(0, 1)$ and U is a uniform random number generator.

Improvise a new harmony Generating a new harmony is called improvisation. A new harmony vector (NHV), $\mathbf{x}' = (x'_1, x'_2, \dots, x'_n)$, is generated based on three rules: (1) memory consideration, (2) pitch adjustment, (3) random selection. In the memory consideration, the value for a decision variable is randomly chosen from the historical values stored in the \mathbf{M} with the probability of HMCR. Every component obtained by the memory consideration is examined to determine whether it should be pitch-adjusted. The *pitch-adjusting process* uses the PAR parameter, which is the probability of pitch adjustment. If it happens that the decision variable x'_i to be pitch adjusted, its value becomes $x'_i \leftarrow x'_i \pm r \times b$ where b is an arbitrary distance bandwidth and $r \sim U(0, 1)$. The main intuition behind the pitch adjustment process is to make a slight modification to the value of chosen variable. The decision variables that are not selected for memory consideration will be randomly chosen from the entire possible range with a probability equal to $(1 - \text{HMCR})$. The adoption of these operations to clustering problem will be described precisely in next sections.

The role of harmony memory \mathbf{M} is to maintain a history of what algorithm learns during the optimization process and exploit this information via memory consideration operation in generating new solutions. The parameter HMCR, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the harmony memory \mathbf{M} , while $(1 - \text{HMCR})$ is the rate of randomly selecting one value from the possible range of values. In other words, HMCR determines the rate of exploration and exploitation in the course of optimization process. A high value for HMCR, forces the algorithm to mostly stick to the exiting solutions in the \mathbf{M} (i.e, exploitation) and consequently leading to less

exploration of the solution space. On the other hand, by choosing a small value for HMCR, the algorithm performs a random behavior in the solution space (i.e., exploration), hence losing all the information collected during the past rounds which deteriorates the effectiveness of the algorithm.

A motivation for pitch adjusting process comes from a theory of the role of sex in evolution (Livnat et al. 2010). Sexual reproduction involves taking half the genes of one parent and half of the other, adding a very small amount of random mutation, and combining them to produce an offspring. The asexual alternative is to create an offspring with a slightly mutated copy of the parent's genes. The pitch adjusting plays the same role as random mutation in sexual reproduction and the PAR parameter controls the amount of the perturbation. It seems plausible that asexual reproduction should be a better way to optimize individual fitness because a good set of genes that have come to work well together can be passed on directly to the offspring. One possible explanation for the superiority of pitch adjusting process is that, over the long term, the criterion for natural selection may not be individual fitness but rather mix-ability of individual solutions. The ability of a set of solutions to be able to work well with another random set of solutions makes them more robust.

Remark 1 In meta-heuristic algorithms diversification refers to a form of randomization in order to explore the search space effectively. In HS algorithm, diversification is essentially controlled by the pitch adjustment and randomization. These are two sub-components for diversification, which might be an important factor for the high efficiency of the HS method. The diversification issue is of more importance if we consider clustering problem which has a more complex solution space and requires an efficient way to effectively explore this solution space. If diversification is strong enough, a great number of zones of the search space may be loosely explored, which will reduce the convergence rate of the algorithm. By contrast, if diversification is kept low in the algorithm design, there is a significant risk of leaving a fraction of the solution space unexplored or even producing far-from-optimal solutions due to trapping in local optima.

Update harmony memory If the new harmony vector, $\mathbf{x}' = (x'_1, x'_2, \dots, x'_n)$, has better fitness function than the worst harmony in the \mathbf{M} , the new harmony is included in the \mathbf{M} and the existing worst harmony is excluded.

The HS algorithm is terminated when the stopping criterion (e.g., maximum NI) has been met. Otherwise, Steps 3 and 4 are repeated.

4.2 The basic harmony search based session clustering

In this section we describe a pure HS based binary session data clustering algorithm dubbed as HSC. In order to cluster sessions using HS algorithm, we must recast the clustering problem as an optimization task that locates the optimal centroids of the clusters. To this end, we define an objective function over the space of properly designated solutions and utilize the HS algorithm to find a solution that minimizes the objection function. In this regard the clustering essentially becomes a search problem over the space of candidate solutions.

The objective function We begin by defining an objective function to capture the goodness of different clustering of binary data points. Although casting the clustering problem as an optimization problem of such an objective function formalizes the problem to some extent, however, we are not aware of any function that optimally captures the notion of a good cluster, since for any function one can exhibit cases for which it fails. Furthermore, not surprisingly, no polynomial-time algorithm for optimizing such cost

functions is known. Therefore, the main challenge in optimization based clustering becomes the formulation of an objective function which is capable of reflecting the nature of the problem so that its minimization reveals meaningful structure (clusters) in the data.

Recall that in session clustering the overarching goal is to partition the set of input sessions $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ into K clusters. Here n is the total number of users and each session $s \in \mathcal{S}$ is represented by a binary vector $\{0, 1\}^m$ over m pages. The i th entry of each session vector indicates whether or not the i th page has been visited by the user. The objective function we utilize to measure the quality of a specific clustering is to minimize intra-cluster similarity while maximizing the inter-cluster similarity. More precisely, the fitness value of each row of \mathbf{M} , which corresponds to one potential solution, is determined by average distance of sessions to the cluster centroid (ADSC) represented by each solution. The fitness value of the clustering solution $\mathcal{C} = (c_1, c_2, \dots, c_K)$ is measured by:

$$\text{fitness}(\mathcal{C}) = \frac{1}{K} \sum_{i=1}^K \left(\frac{\sum_{s \in S_i} \mathcal{D}(s, c_i)}{n_i} \right), \tag{1}$$

where K is the number of clusters, c_i is the centroid of the i th cluster, S_i is the set of sessions assigned to i th cluster, n_i is the number of sessions in S_i , and $\mathcal{D}(\cdot, \cdot)$ calculates the distance between two binary vectors. In this function the best clustering tries to minimize the between cluster similarity while increases inner cluster similarity. So the best clustering is one with least ADSC value.

Representation of solutions Another choice of decision we have to make in order to apply HS algorithm is the representation of solutions. In our representation, each row of \mathbf{M} is comprised of n elements where i th element is an integer in $\{1, 2, \dots, K\}$ and corresponds to the cluster the i th session s_i is assigned to. Putting another way, one can think of each row as the assignment vector $\mathbf{a} \in \{1, 2, \dots, K\}^n$, where a_i represents the cluster number the i th session is assigned to.

The assignment vector \mathbf{a} has the property that each session must be assigned exactly to one cluster. We also note that the assignment vector \mathbf{a} must represent a valid clustering. A clustering is valid if for every cluster there is at least one session assigned to that cluster. These two properties must be preserved in applying the HS operations to clustering solutions. The clustering associated with a row in \mathbf{M} is represented by $\mathcal{C} = (c_1, c_2, \dots, c_K)$ where c_i corresponds to the centroid obtained from the i th block.

```

Algorithm 3. Generate NHV

for  $i = 1, 2, \dots, n$ 
  Generate an uniform random number  $r \sim U(0, 1)$ 
  if  $r > \text{HMCR}$  then
    Sample  $j$  uniformly at random from  $\{1, 2, \dots, \text{HMS}\}$ 
    Set  $\text{NHV}[i]$  to be  $\mathbf{M}[j][i]$ 
    Apply PAR process to  $\text{NHV}[i]$ 
  else
    Randomly choose a value in  $\{1, 2, \dots, K\}$  for the  $\text{NHV}[i]$ 
  end if
end for
Check/Modify the solution to be a valid clustering
    
```

Improvisation of new solutions Having defined the representation of solution and an appropriate fitness function to evaluate different solutions, the next step is devise an efficient solution from the existing solutions in the harmony memory \mathbf{M} . This new vector is generated by sampling from the \mathbf{M} or choosing a random assignment.

The detailed steps of the process of generating new solution is outlined in Algorithm 3. In this algorithm n and K are the number of training sessions and clusters, respectively. Following the representation schema discussed above each solution in \mathbf{M} is composed of n decision variables and consequently the improvisation process runs for n iterations. At i th iteration we sample a random number r uniformly in $[0, 1]$ to decide the next step. If r is larger than HCMR, then the i th component must be selected from the existing solutions in \mathbf{M} (i.e., exploitation phase) otherwise it will be selected uniformly random from $\{1, 2, \dots, K\}$ (i.e., exploration phase). A remarkable strength of HS hinges on its improvisation operators, which play a major role in achieving a good trade-off between exploration and exploitation.

In exploitation phase, when the decision about the value of a particular component is made from the \mathbf{M} , the value is subject to the pitch adjusting process. The goal of pitch adjusting process is to make a small perturbation in the value of decision variable. For continuous valued optimization problems the pitch adjusting process is straightforward as we need to add or subtract a small random value from the current value of the decision variable. But for discrete valued optimization problems such as the clustering we are encountering in this paper, the pitch adjustment needs to be performed carefully. This is because in discrete valued problems, the small perturbation in the value of a decision variable is not well-defined.

To propose an efficient pitch adjusting mechanism for clustering problem we proceed as follows. Let $\mathbf{d} = (\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_K) \in \mathbb{R}_+^K$ be the distance of a session to the K centroids obtained by a row of \mathbf{M} (the exact computation of this vector will be clear in what follows). Let $\mathbf{p} = \mathbf{d} / \|\mathbf{d}\|_1$ denote the corresponding probability vector over simplex where $\|\cdot\|_1$ is the ℓ_1 norm of a vector. Equipped with this probability vector, now we introduce the proposed pitch adjusting process. In particular, when a value of a solution is subject to pitch adjustment, its current cluster is replaced with another cluster $j \in [K]$ with probability p_j . This is reminiscent of pitch adjusting process for continuous valued variables since it changes the current cluster to another cluster promotional to the distance of the session to clusters.

To better illustrate Algorithm 3, Fig. 2 shows the steps of generating a new harmony vector for a simple clustering example with five sessions and four clusters. Assume Fig. 2(I) represents the initialization of \mathbf{M} . NHV in Fig. 2 shows the novel solution that must be generated from the existing solutions in \mathbf{M} . For simplicity, we suppose that all choices are made from the \mathbf{M} (e.g., HMCR = 1) and the effect of PAR parameter (i.e., the probability that the chosen value is subject to pitch adjustment) is ignored in the selection of parents. A marker exists attributed to each row in \mathbf{M} that is initialized to one and represents the potential entry which can be selected to be included in the NHV as shown in Fig. 2(II), (III). Figure 2(IV)–(V) show the remaining steps to complete the generation of NHV. We note that in final stage, since the newly generated solution has a better fitness value than the solution in the third row of the \mathbf{M} , it is replaced with this solution as shown in Fig. 2 (V).

Representation of centroids We now turn to describing an effective way to compute the centroid of each cluster. A naive idea would be to average the sessions assigned to each cluster as the cluster centroid. But since the session data are binary vectors, this representation significantly deteriorates the quality of clustering. Hence, we propose a novel representation that is more appropriate for binary data points. The main idea is to find a binary vector which is the most representative of the vectors inside the cluster. We turn this idea to a combinatorial optimization problem and propose an algorithm to find it efficiently.

Let $\mathbf{a} \in \{1, 2, \dots, K\}^n$ be a row in \mathbf{M} and $S = S_1 \cup S_2 \cup \dots \cup S_K$ be the corresponding partitioning of sessions into K partitions. The goal is to compute the set of centroids $\mathcal{C} = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K)$ where \mathbf{c}_i is the centroid corresponds to i th partition $S_i, i \in [K]$. Since each partition is a set of binary vectors, it is not feasible to directly follow the averaging method that is the mostly used mechanism is clustering non-binary vectors. Therefore, we introduce a novel idea to find the representative of a set of binary vectors. To this end, let $S' = \{s_1, s_2, \dots, s_q\}$ be a set of binary vectors where $s_i \in \{0, 1\}^m$. The goal is to find a binary vector $\mathbf{c} \in \{0, 1\}^m$ which is the most representative binary vector for the vectors in S' . A simple idea is to solve the following optimization problem

$$\begin{aligned} &\text{minimize}_{\mathbf{s}} && \frac{1}{q} \sum_{i=1}^q d_H(\mathbf{s}, s_i) \\ &\text{subject to} && \mathbf{s} \in \{0, 1\}^m, \end{aligned}$$

where $d_H(\cdot, \cdot) : \{0, 1\}^m \times \{0, 1\}^m \mapsto \mathbb{N}$ is the Hamming distance between two binary vectors. Since this problem is computationally hard to optimize, we propose an efficient

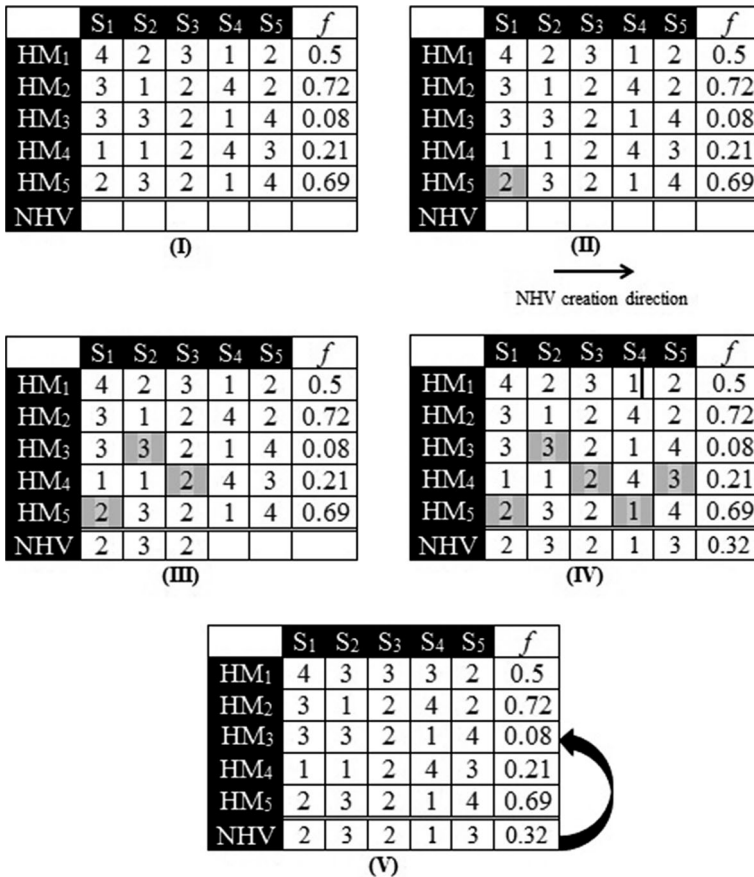


Fig. 2 A simple example to illustrate the new harmony vector (NHV) generation process. The example has five sessions and the size of \mathbf{M} is set to be five

approximate solution. To do so, let $\widehat{\mathbf{s}}$ be the average of solutions contained in partition \mathcal{S}' as computed below:

$$\widehat{\mathbf{s}} = \frac{1}{|\mathcal{S}'|} \sum_{\mathbf{s} \in \mathcal{S}'} \mathbf{s}.$$

Since $\widehat{\mathbf{s}}$ is the average of solutions in \mathbb{R}^p , it is the best representative of candidates in \mathcal{S}' but it is not a binary vector. To obtain a binary representation of $\widehat{\mathbf{s}}$, we solve the following optimization problem:

$$\begin{aligned} \operatorname{argmax}_{\mathbf{b}} \quad & \frac{\mathbf{b}^\top \widehat{\mathbf{s}}}{\|\mathbf{b}\|_2} \\ \text{subject to} \quad & \mathbf{b} \in \{0, 1\}^m, \end{aligned} \quad (2)$$

Also this is an integer programming problem but fortunately its global maximum can be found very efficiently. To see this, we propose the following algorithm and show that it computes the optimal solution to integer programming problem. The algorithm first sort the entries in $\widehat{\mathbf{s}}$ in descending order. Let $\widetilde{\mathbf{s}} = \widetilde{s}_{(1)}, \widetilde{s}_{(2)}, \dots, \widetilde{s}_{(m)}$ be the ordered vector. The algorithm runs in m iterations. At each iteration i , we form a binary vector \mathbf{b}_i whose elements are 1 for the i largest positions in $\widetilde{\mathbf{s}}$ and 0 otherwise. Then we compute the weight of i th binary vector as

$$w_i = \frac{\widetilde{\mathbf{s}}^\top \mathbf{b}_i}{\|\mathbf{b}_i\|_2} = \frac{\sum_{j=1}^i \widetilde{s}_{(j)}}{\sqrt{i}}.$$

The final solution is a m -dimensional binary vector that attains the maximum weight over all m iterations, i.e., $\mathbf{b}_* = \operatorname{arg max}_{i \in \{1, 2, \dots, m\}} w_i$. We now show that \mathbf{b}_* is the best binary vector representing the average of candidates in \mathcal{S}' . Since \mathbf{b}_* is a m -dimensional binary vector, its norm $\|\mathbf{b}_*\|_2$ can have at most m different values $\{1, \sqrt{2}, \dots, \sqrt{m}\}$. To see \mathbf{b}_* is the optimal solution to (2), we consider the solution of (2) for different values of the norm of the optimal solution. For a binary vector $\|\mathbf{b}_i\|$ with norm \sqrt{i} , the optimization problem in (2) attains its maximum by setting to one the entries of $\|\mathbf{b}_i\|$ which corresponds to the i largest entries of $\widetilde{\mathbf{s}}$. Since $\|\mathbf{b}_*\|$ can take only m distinct values, we go through all these values and find the best \mathbf{b} for each iteration. Since each iteration i works with i largest elements from $\widetilde{\mathbf{s}}$, the sorting at the beginning helps us to consider only first i element at i th iteration.

4.3 Hybrid sequential session clustering

The HSC algorithm performs a globalized searching for solutions, whereas the traditional k -means clustering procedure performs a localized searching. In localized searching, the solution obtained is usually located in the proximity of the solution obtained in the previous step. The k -means clustering algorithm uses the randomly generated seeds as the initial clusters centroids and refines the position of the centroids at each iteration. The refining process of the k -means algorithm indicates that the algorithm only explores the very narrow proximity, surrounding the initial randomly generated centroids and its final solution depends on these initially selected centroids.

So, HSC and k -means algorithms have complementary strong and weak points. HSC is good at finding promising areas of the search space, but not as good as k -means at fine-tuning within those areas. On the other hand, k -means algorithm is good at fine-tuning, but

lack a global perspective. It seems a hybrid algorithm that combines HSC with k -means can result in an algorithm that can outperform either one individually. For this reason, we present hybrid clustering approaches that uses k -means algorithm to replace the refining stage in the HSC algorithm.

The hybrid algorithms combine the power of the HSC algorithm with the speed of k -means and the global searching stage and local refine stage are accomplished by those two modules, respectively. Roughly speaking, the HSC finds the region of the optimum, and then the k -means takes over to find the optimum centroids. We need to find the right balance between local exploitation and global exploration.

We present two different versions of the hybrid clustering, depending on the stage when we carry out the k -means algorithm.

4.3.1 Harmony k -means session clustering

The first hybrid algorithm, namely HKSC, the HSC algorithm performs a coarse search and then utilizes the k -means algorithm as a local improver to improve the obtained solution. In this procedure HSC first produces solutions within a fixed T number of iterations and then, the best generated cluster centers are passed to k -means algorithm as initial centers.

The main intuition for the HKSC algorithm stems from the the observation that the performance of the k -means algorithm, despite its fast convergence and fine-tuning capabilities, is sensitive to the selection of the initial cluster centroids. The k -means algorithm performs well when provided with initial centers which are somehow good representative of optimal clusters. To illustrate this fact, we provide a simple setting which shows this sensitivity and sub-optimality of the k -means method. To do so, consider a clustering problem over real line with five clusters with centers $\mathcal{C} = \{c_1, c_2, \dots, c_5\}$. For the simplicity of exposition we assume that $c_1 < \dots < c_5$ and every two consecutive centers are located in a distance of Δ . We assume that there is a ball of radius δ around each center and n data points are distributed in these balls uniformly at random. Hence for the optimal clustering of these n points, the sum of squared Euclidean distance of points to cluster centers is $O(n\delta^2)$, because the distance of each point to its cluster center is at most δ .

Algorithm 4. Harmony k -means Session Clustering (HKSC)

```

Initialize harmony memory M
while the stopping condition is not met do
  Create a new harmony vector
  Cluster users according to the vector
  Assess the cluster's ADSC measure
  Replace this vector with the worst one in M
end while
Pass the best centroid to  $k$ -means as the initial centroid for  $k$ -means
while the stopping condition is not met do
  Assign all the remaining users to their most similar cluster according to centroids for the clusters
  Update the centroids
end while

```

We utilize the k -means algorithm to cluster these points. To this end, we initialize the k -means algorithm by choosing five data points at random as the initial centers of the clusters. There is a chance that no data point from first cluster, two data points from the third cluster, and on data point from the remaining clusters have been chosen a centers. In the first round of k -means, all points in clusters 1 and 2 will be assigned to the cluster

centered at c_1 . The two centers in cluster 3 will end up sharing that cluster. And the centers in clusters 4 and 5 will move roughly to the centers of those clusters. Thereafter, no further changes will occur. This local optimum has cost $\Omega(n\Delta^2)$. We note that this cost can be made arbitrarily far away from the optimum cost by setting the distance between the consecutive centers, i.e., Δ , large enough.

The above example illustrates that, despite the convergence of k -means algorithm to local minimum, the initialization of k -means algorithm significantly affects the final result. Therefore, we hinge on the explorative power of the HSC algorithm to find the area of solution space with good clustering results and feed the k -means with a solution from this area for further improvement. The detailed steps of the hybrid HKSC algorithm are shown in Algorithm 4. The algorithm proceeds in two stages. At the first stage the a clustering of data points is obtained similar to the HSC algorithm. When the HSC is completed or shows a negligible trend of improvement after T iterations, the k -means begins its job. The best generated cluster centers from the harmony memory \mathbf{M} are passed to k -means as the starting point.

4.3.2 Interleaved harmony k -means session clustering

In the HKSC algorithm, good initial clustering centroids are obtained using HSC algorithm and then the k -means method is utilized to refine the clustering centroids to further improve the objective function. Our second hybrid session clustering algorithm is an iterative version of the HKSC method. The basic idea is to integrate the local k -means method into the HSC to gradually refine the solutions in the course of optimization process.

In the interleaved HKSC (IHKSC) hybridized algorithm, the clustering alternates between k -means and HSC algorithms for a fixed number of iterations. In particular, after every fixed number of iterations, say L , the k -means uses the best vector from the harmony memory \mathbf{M} as its starting point to the next run. Then \mathbf{M} is updated if the locally optimized vectors have better fitness value than those in HM and this procedure repeated until stopping condition. In contrast to HKSC algorithm where the refinement is only conducted on the final solution obtained by the HSC, in IHKSC every generated solution is subject to the refinement. The integration of k -means into HSC has the effect of slowing down the algorithm so that instead of pure exploration of the solution space, it has the tendency to infrequently refine the intermediate solutions, unless there is a compelling reason to switch to another area in the solution space with better clustering results.

5 Time complexity analysis

In this section we investigate the timing analysis of the proposed algorithms. The analysis is performed for the both HS-based and hybrid algorithms. The notation used in this section are defined as follows:

- d : number of dimensions that satisfies the condition of MinFreq.²
- r : number of pages that are recommended to a user.
- T_{hs} : Number of rounds that HS iterates.
- T_{km} : number of rounds that k -means iterates.

² MinFreq: is the short form of Minimum Frequency, which means those dimension of vectors are used for calculating the distance between two given vectors, which are bigger than MinFreq.

- HMS: size of harmony memory.
- K : number of clusters.
- n : number of sessions.
- T_{it} : number of rounds that the interleave algorithm iterates.
- g : number of generations that recommender makes recommendations.
- m : number of pages.

Lemma 1 *The time complexity of HSC is $O(T_{hs} \times n)$.*

Proof Within a given iteration, when a solution is created (NHV), the components should be filled for n number of times. Therefore a given iteration is dependent on the number of session, with the timing complexity of $O(n)$. The algorithm iterates for T_{hs} number of times, that makes the timing complexity as claimed.

Lemma 2 *The time complexity of KSC is $O(T_{km} \times d \times K \times n)$.*

Proof The k -means algorithm complexity mainly depends on the number of clusters, $O(K)$. As the number of clusters increases the algorithm needs more time to cluster the data. if the dimension of two given vectors are considered that meet the requirement of MinFreq, there will be d number of dimensions that the algorithm considers for measuring closeness of two given vectors. Therefore k -means time complexity depends on three factors of dimensions of data, number of clusters and number of sessions to be clustered. The time complexity of each iteration is $O(d \times K \times n)$. The algorithm iterates for a given number of rounds that makes the time complexity as $O(T_{km} \times d \times K \times n)$.

Lemma 3 *The time complexity of HKSC and IHKSC are $O(T_{hs} \times T_{km} \times d \times K \times n^2)$ and $O(g \times T_{hs} \times T_{km} \times d \times K \times n^2)$, respectively.*

Proof The time complexity of HSC for a single iteration $O(n)$. On the other hand the time complexity of k -means for one generation is $O(K \times n)$. Therefore HKSC is executed in one iterations with the time complexity of $O(K \times n^2)$. Also, the algorithm needs to consider d number of dimensions that satisfy the condition of MinFreq. Therefore if HSC and KSC iterate for T_{hs} and T_{hs} , respectively, the overall duration of the algorithm is as $O(T_{hs} \times T_{km} \times d \times K \times n^2)$. If HKSC iterates in an interleave form for g number of generations, then the time complexity of IHKSC becomes $O(g \times T_{hs} \times T_{km} \times d \times K \times n^2)$ as desired.

Lemma 4 *The time complexity of the recommender system is $O(\text{SessionClustering}) + O(g \times m \times r)$.*

Proof In a single generation of recommendation the cluster of an active session should be identified. After each selection the user is re-clustered. The re-clustering requires comparison between two vectors of an active session and a centroid. Therefore this comparison is dependent on the number of pages in the system, with time complexity of $O(m)$. When the cluster of an active session is identified then the most similar pages to the ones that the user has visited so far should be recommended. This step will make the timing analysis as $O(r \times m)$. If the recommendation iterates for g generations, then the time complexity will be $O(r \times m \times g)$. Before executing the main recommendation procedure it is required that the sessions to be clustered. Depending on the clustering algorithm used, its timing complexity will be added to the recommender system's timing analysis.

6 Experiments

In this section, experiments are carried out to evaluate how effective the proposed algorithms are. Different assessment measures have been used and the dynamic parameters of the algorithms are fine-tuned. The proposed clustering and recommendation algorithms are compared against well-known algorithms in this area. In summary the following key experiments have been conducted:

1. *Considering different assessment measures* to better reveal the defects and significances of the proposed variations we used two assessment measures, namely visit coherence (VC) and ADSC for cluster analysis and three other measures, namely accuracy, coverage and F-measure, for recommender performance assessment.
2. *Precise fine-tuning process* the performance of HS can heavily be dependent on its dynamic parameters. To assure reaching the best possible performances, we considered three scenarios that each will fine tune a specific parameter. We also investigate the influence of the window size w and the number of cluster K on the performance of recommendation algorithms to decide the best settings of these parameters.
3. *Rigorous clustering comparisons* we have compared the proposed session clustering algorithms with different graph based, k -means and genetic algorithm based clustering algorithms.
4. *Rigorous recommender comparisons* we have compared the proposed recommender algorithms with graph based, learning based, and other algorithms that can not be categorized as graph based or learning based algorithms. The proposed algorithms are also compared with recommender algorithms based on optimization based methods including genetic algorithm and k -means, as well.

6.1 The CTI dataset

The method described in this paper can be applied to usage data of any Web site. We have used the access logs of DePaul University CTI Web server³, based on a random sample of users visiting the site for two weeks period during April 2002 (DePaul Web Server Data). This dataset contains 13,745 distinct user sessions of length more than one over 683 distinct pages. The only cleaning step performed on this data was the removal of references to auxiliary files (e.g., image files). No other cleaning or preprocessing has been performed in the first phase. The data is in the original log format used by Microsoft IIS. Each session contains a sequence of pages along with their weights (durations a user spent on the page). We split the dataset in two non-overlapping time windows to form training and test datasets. Randomly, 70 % of the dataset selected for training and excluded from the main dataset, while another 30 % is used for testing.

Figure 3 shows the relationship between the number of pages and the number of sessions. Most of the sessions have pages between one to ten while a minority of the sessions have pages between 51 and 60. By increasing the number of pages, the number of sessions decline. This point indicates that most of the sessions in this dataset have short lengths in terms of visited pages. As the number of sessions have pages within (1–10) is very high comparing to other categories. Figure 4 details the distribution of pages in this category [i.e., in the interval (1–10)]. We considered sessions with the length of one as noise and

³ <http://maya.cs.depaul.edu/classes/ect584/data/cti-data.zip>.

then eliminated these sessions to prevent from interruption of noise in our processing. According to Fig. 4, most of the sessions have the length of two.

6.2 Measures

We now turn to explaining the measures that are used for assessing the quality of the generated clusters and the quality of the recommendations. For evaluation of session clustering algorithms two different measures of ADSC and VC were selected. To assess the performance of the RS we utilize accuracy and coverage along with F-measure. A brief explanation of these measures is given in the following subsections.

Clustering quality measures In order to assess the quality of the generated clusters two measures namely ADSC and VC are used. The ADSC measure for a clustering solution with the set of centroids given as $\mathcal{C} = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K)$ with the corresponding partitioning of sessions as $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \dots \cup \mathcal{S}_K$ is given by:

$$ADSC = \frac{1}{K} \sum_{i=1}^K \left(\frac{\sum_{s \in \mathcal{S}_i} D(\mathbf{c}_i, \mathbf{s})}{|\mathcal{S}_i|} \right), \tag{3}$$

where K is the number of clusters and function $D(\cdot, \cdot)$ calculates the distance between \mathbf{c}_i and a session \mathbf{s} . In this function the best clustering aims to minimize the between cluster similarity while maximizing the inner cluster similarity.

The other measure we use to evaluate the quality of a clustering algorithm is the VC (Jalali et al. 2010), defined by:

$$\alpha = \frac{\sum_{i=1}^n \beta_i}{n}, \text{ with } \beta_i = \frac{|p \in \mathcal{S}_i \cap p \in \mathbf{c}_i|}{|\mathcal{S}_i|}.$$

The VC measure α is the percentage of VC that should be considered for different values of MinFreq.

Recommendation quality measures To evaluate the performance of recommendations, we use the typical quality measures: accuracy and coverage, two well accepted performance indicators in the information retrieval field. Given a session $\mathbf{s} \in \{0, 1\}^m$ in the testing set and a window size w (the portion of an active session used to produce recommendations), the algorithm recommends a subset of pages $\mathbf{r} \subseteq \mathcal{P}$ only based on the w pages from the session \mathbf{s} . The concept of window makes the system capable of providing recommendations for users at various steps of browsing. Then the accuracy and coverage measures are defined as:

$$\begin{aligned} \text{accuracy}(\mathbf{s}, \mathbf{r}) &= \frac{|\mathbf{s} \cap \mathbf{r}|}{|\mathbf{r}|}, \\ \text{coverage}(\mathbf{s}, \mathbf{r}) &= \frac{|\mathbf{s} \cap \mathbf{r}|}{|\mathbf{s}|}. \end{aligned}$$

The F-measures combines these two quality measures in a single measure defined as:

$$F - \text{measure}(\mathbf{s}, \mathbf{r}) = 2 \times \frac{\text{accuracy}(\mathbf{s}, \mathbf{r}) \times \text{coverage}(\mathbf{s}, \mathbf{r})}{\text{accuracy}(\mathbf{s}, \mathbf{r}) + \text{coverage}(\mathbf{s}, \mathbf{r})}.$$

As it was mentioned before, two types of recommendations are used. One is based on the minimum frequency of each dimension of the vectors and the other is based on the number of recommendations that can be made.

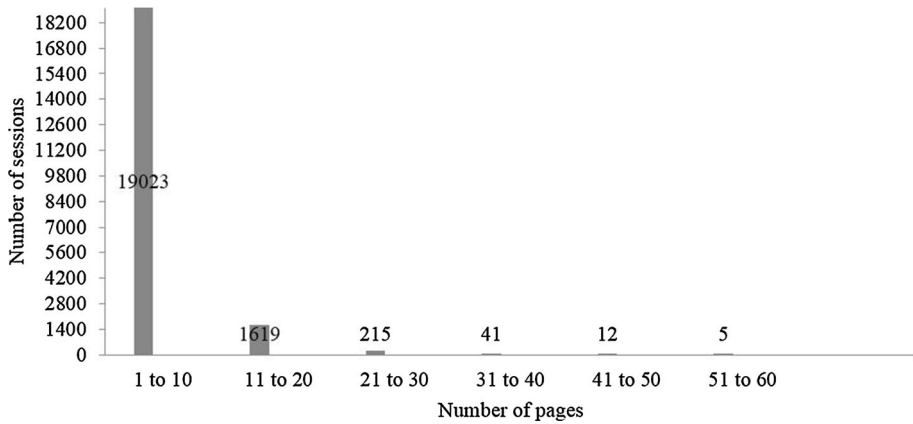


Fig. 3 Percentage of sessions and pages in the CTI dataset

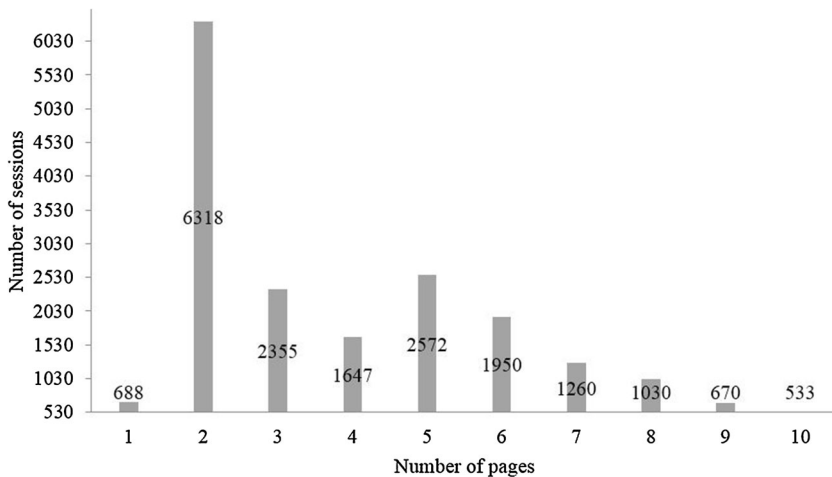


Fig. 4 Ratio of sessions with 1–10 pages in the CTI dataset

6.3 Baseline algorithms

In this section we briefly review the baseline algorithms. We utilize the proposed HSC, HKSC, IHKSC clustering algorithm in there different RS termed as **HSCR**, **HKSCR**, and **IHKSCR**, respectively. We selected a variety of recommendation algorithms exploiting different techniques for Web page recommendation task. In particular, we intend to compare the proposed recommender algorithms with the following baselines:

1. Distributed learning automata based recommender (DLA) (Forsati and Meybodi 2010): This algorithm employs the Web usage data and underlying site structure to recommend pages to the current user. In DLA algorithm, the transition matrix and personalization vector in the original PageRank algorithm are computed based on usage data instead of link structure. For this reason, a distributed learning automata

- learns the transition probability matrix from the behavior of the visiting users which are available in the site's log file. In addition, the personalization vector is computed based on the visiting rate of pages, preferring pages which are visited by more users. Having the transition matrix and personalization vector which obtained from the knowledge acquired from previous users' visits, the PageRank algorithm is used to compute the rank of each page and recommend pages to the current users.
2. **Weighted association rule mining based recommender (WARM)** (Forsati and Meybodi 2010): In this algorithm first, the weighted association rules are extracted from the log data and similarity between active user session is calculated upon the weighted rules instead of an exact match for finding the best rule. Finally, the recommendation engine will then find the most similar rules to the active user's session with the highest weighted confidence by scoring each rule in terms of both its similarity to the active session and its weighted confidence.
 3. **DLA + WARM** (Forsati and Meybodi 2010): This algorithm is a hybrid algorithm based on distributed learning automata and the weighted association rule mining algorithm, which solves the problem of unvisited or newly added pages. The algorithm has 4 steps as follows: (1) cluster the pages based on users usage pattern; (2) generate the seed recommendation set; (3) extend the seed set by clusters to generate the candidate set, and (4) apply the HITS algorithm to rank the candidate set and generate final recommendation set.
 4. **Web-based recommendation system to predict user future movements (WebPUM)** (Jalali et al. 2010): The WebPUM is an algorithm for RS based on Web usage mining architecture. The algorithm has two interleaved phases; off-line and online. The algorithm generates navigation patterns in the off-line phase by utilizing graph partitioning algorithm. An online phase within the prediction engine has the task to predict user future requests. Classifying the user current activities based on navigation patterns in a particular Web site is the main objective of this phase. In addition, creating a list of recommended Web pages as prediction of user future movement is another objective in this phase. To predict the future activities of the users the authors applied the LCS algorithm.
 5. **Dynamic personalization of Web Sites (SUGGEST)** (Baraglia and Silvestri 2007): The SUGGEST is a Web page recommendation algorithm that provides useful information to make easier the navigation of users and to optimize the Web server performance. This algorithm has a two-level architecture composed by an off-line creation of historical knowledge and an online engine that understands users' behavior. The system works in the way that when the requests arrive at the system, it incrementally updates a graph representation of the Web site based on the active user's session. A graph partitioning algorithm is used to classify the active sessions using a graph partitioning algorithm.
 6. **Improved profile aggregation based on clustering of transactions (IPACT)** (AlMur-tadha et al. 2011): This algorithm is a two phases recommender algorithm. In the first phase (offline phase) the algorithm is responsible for partition the filtered transactions into clusters of similar pageviews. Then, the future web navigation profiles are created according to these clusters using PACT methodology. The input to the offline phase is the preprocessed web server logs file and the outputs are clusters of navigation transactions and the web navigation profiles. The second phase (online phase) has the responsibility of matching the new user transaction (current user session) to the profile shares common interests to the user. The inputs to this step are the web navigation profiles generated from the offline phase and the current user session. The output will

be the recommendation set in addition to the best profile that matches the user interests.

7. LCS (Jalali et al. 2009): This algorithm consists of two main phases. Navigation patterns mining has been based on the graph partitioning algorithm as offline phase of this algorithm. In order to classify an active user's session, the navigation patterns that include the larger number of similar Web pages in the session are looked. To this end, pattern search approaches can be utilized to find similar Web pages between the current active session and navigation patterns. In this paper, the LCS algorithm is used to find the longest subsequence common to measure the similarity of sessions. In the online phase, the RS predicts a set of Web pages as user next intentions.
8. Association rule mining based recommender (ARM) (Mobasher et al. 2001): This algorithm uses association rule mining to generate navigational patterns in the off-line phase. In the online phase, recommendation sets are selected from the navigational models, based on a localized degree of hyperlink connectivity with respect to a users current location within the site.
9. Learning automata and graph partitioning based recommender (LAGP) (Mehr et al. 2011): This algorithm recommends new Web pages to a user based on the navigation history of the user, the usage data and the hyperlink graph of the web site. The hyperlink graph of a web site is partitioned by a graph partitioning algorithm. Then a distributed learning automaton is used to determine similarities of the Web pages based on the assumption that two consecutive visited pages are similar if they have been partitioned into the same partition. Afterwards, the similarity between Web pages is used to build a Markov model of the users transitions in the web site. Finally, this Markov model is used to predict the probability of visiting new pages based on which Web pages are recommended to the users.
10. Dynamic cellular learning based recommender (DCLA) (Talabeigi et al. 2010a): This algorithm recommends Web pages based on cellular learning automata. The DCLA algorithm uses Web usage data and structure of the Web site to learn user navigation patterns and predicting user's future requests. Each Web page is represented by a cell and equipped with a learning automaton. In this model, each cell has $n - 1$ neighbors and each learning automaton has $n - 1$ actions where n is the number of Web pages. The task of each learning automaton is to learn to making better recommendation (i.e, the best action).
11. Genetic algorithm based recommender (GAR) (Krishna and Murty 1999): This algorithm partitions the session data into a fixed number of clusters by using the genetic optimization algorithm and utilizes the partitioned sessions to make recommendations. The genetic operators that are used in this algorithm are the selection, cut and splice crossover policy and the distance based mutation. The representation of GA is to consider a chromosome of length n and allow each allele in chromosome to take values from 0 or 1. The crossover probability (pc) is 0.6, mutation probability (pm) is 0.1 and genetic pool size is as same as HMS in HS. When the genetic algorithm is utilized only for clustering we use **GA** to refer to it.
12. KSR: This basic algorithm uses the k -means algorithm to partition the session data and utilizes the obtained clusters for recommendation. We use **KSC** when we investigate the quality of clusters generated by k -means algorithm.

6.4 Empirical study of parameters

We now turn to investigating the performance of the proposed RS under different settings of dynamic parameters to find out the best configuration for our experiments.

6.4.1 Empirical study of HS parameters

In this subsection, the evaluation of algorithms are investigated under different settings of three important parameters, the HMS, HMCR, and PAR. ADSC is used as the main measure of performance assessment, as shown in (3), since it can be a good indication to see if the clusters are fully separated. To study the impact of changing various parameters we used all the proposed variations of clustering algorithms, i.e., **HSC**, **HKSC**, and **IHKSC**. Furthermore different scenarios are considered where in each of them just one parameter is changed while fixing other parameters. These scenarios are shown in Table 4.

In Fig. 5, we depict the results of the first scenario, corresponding to the Experiment (I) in Table 4 for three algorithms. For all of the algorithms the best result gained when the HMS parameter is set to five. In **IHKSC**, the best results were revealed for HMS with values of five and eight, indicating that varying the the HMS in this range does not make any further changes to the search strength of **IHKSC** algorithm. The other obvious fact that can be inferred from the results depicted in Fig. 5 is that by increasing the HMS value, performance of the algorithms in terms of ADDC declines.

In Fig. 6, the second scenario is tested on the three variations proposed in this paper, to see the impact of HMCR parameter. As mentioned earlier, HMCR determines the rate of choosing one value from the historical values stored in the harmony memory **M**, which is referred to as exploitation parameter. The larger HMCR is, the less exploration will be achieved and the algorithm further relies on the stored values in harmony memory and this potentially leads the algorithm getting stuck in a local optimum. On the other hand, choosing HMCR too small will decrease the algorithm efficiency and the HS algorithm behaves like a pure random search, with less assisting from historical data saved in the memory. As it can be seen in Fig. 6, ADSC value has a downward trend. It means that selecting HMCR value large here could improve the performance, since it relies more on historical data, preserved inside the harmony memory, and the historical data were effective enough in conducting the procedure to find the best possible solution.

Most of the published HS-based applications such as Forsati et al. (2008) have used a range of values between 0.5 and 0.95 for HMCR, and our results show that **HSC** has the best performance in terms of ADSC value when HMCR is set to 0.95. In the two other algorithms of **HKSC** and **IHKSC**, a similar behavior can be observed.

In the last set of the experiments, as shown in Table 4, HMCR and HMS are set to 0.95 and 5, respectively and the ADSC of the proposed algorithms for different values of PAR

Table 4 Different scenarios considered for testing dynamic parameters

Scenario	Fixed configuration	Variable parameter	Value
Experiment (I)	HMCR = 0.95, PAR = 0.51, K = 40	HMS	5, 8, 15, 20 ($=\frac{K}{2}$), 40 ($=K$), 45
Experiment (II)	HMS = K, PAR = 0.5, K = 40	HMCR	0.05, 0.1, 0.15, \dots , 0.95
Experiment (III)	HMS = 5, HMCR = 0.95, K = 40	PAR	0.09, 0.078, 0.061, 0.051

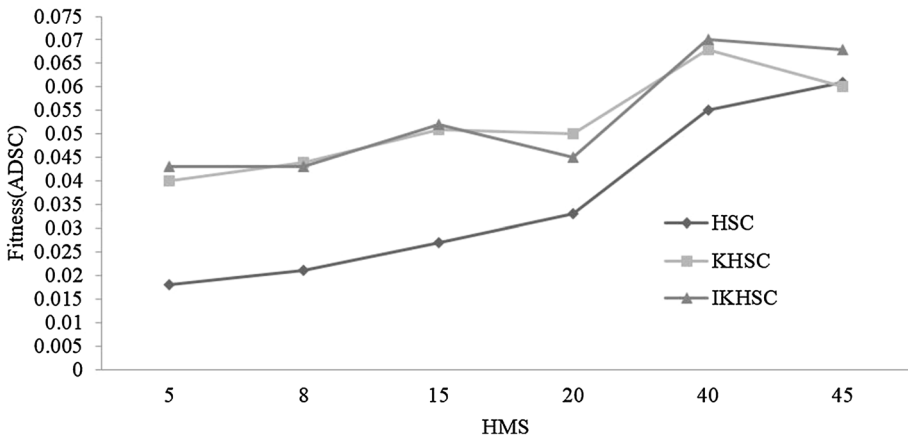


Fig. 5 Fine-tuning of the size of the harmony memory HMS [i.e., Experiment (I) in Table 4]

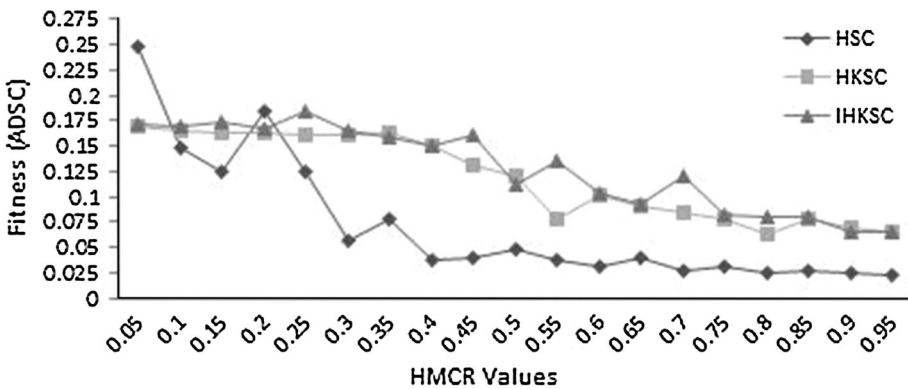


Fig. 6 Fine-tuning of the HMCR parameter [i.e., Experiment (II) in Table 4]

parameter is evaluated. As it can be inferred from the results in Fig. 7, the performance of **HSC** and **HKSC** do not have significant changes as the value of PAR parameter increases or decreases. In **HSC**, when PAR is 0.09 and 0.078, the same result gained while this amount had a nominal changed when PAR = 0.061 and 0.051. Furthermore, it can be inferred that, choosing a PAR value in [0.05, 1) interval can not make huge difference in the final results. In other words, this interval is reliable enough in terms of selecting different values, that results in the same or very similar fitness (ADSC) values.

Based on the above results, the optimal configuration of dynamic parameters in the rest of experiments are summarized in Table 5.

6.4.2 Empirical study of number of clusters

In Fig. 8, we investigate the effects of number of clusters on ADSC measure. In order to deal with randomization nature and its effects on clustering quality, we set the number of

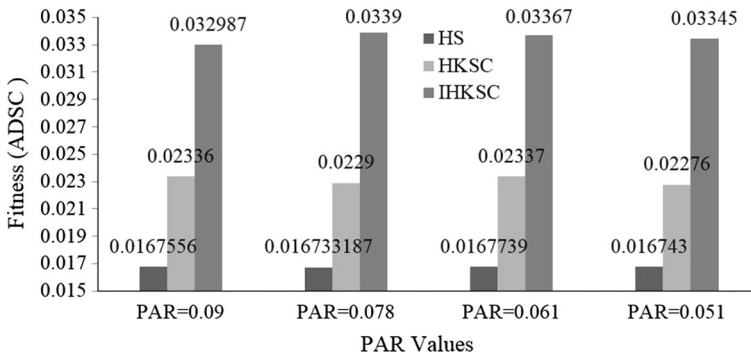


Fig. 7 Fine-tuning of the PAR parameter that [i.e., Experiment (III) in Table 4]

iterations of both HS based and k -means algorithms to 500 number of executions. According to Fig. 8, in **HSC** and the hybridized variations the number of suitable clusters vary. This difference is the result of hybridization of HS with k -means. In **HSCR** the best number of clusters is 60 while in **HKSC** and **IHKSC** the best number of cluster is 50. Hence different settings are used for global and local search algorithms.

6.4.3 Impact of active window size on recommendation

Now we investigate the effect of active window size w on the accuracy of recommendation algorithms. Window size is the number of pages that a user visits before receiving recommendations and choosing its right size is critical for the performance of Web page recommendation algorithms. On the one hand, considering low values for window size it is likely to lose track of user’s interest. On the other hand, setting high values for window size will also mislead the recommendation process, since the user’s navigational pattern may change during a specific time. Therefore, we conduct an empirical study to decide the best value of window size for the CTI dataset in our experiments.

To this end, we considered typical windows sizes as used in different papers (Forsati and Meybodi 2010; Talabeigi et al. 2010a), ranging from 1 to 4. The experiments are carried out independently for each algorithm, as shown in details in Figs. 9, 10, 11 and 12. For each value of window size $|w|$, by increasing the number of recommended pages (RecPgs), the overall accuracy declines as more pages are suggested while less number of pages are included in the recommended subset. In simpler terms the ratio of user visits and the recommended pages declines.

In Fig. 13 an overview of different values of window size for the three proposed recommenders are shown. According to Fig. 13, selection of very low values for window size will lead to poor performances, while performance of the RS for $w = 4$ shows at its peak. Therefore we suggest utilization of $w = 4$ but to have justified comparisons with other works we used $w = 3$. It is noticeable that when the window size is $w = 3$, there is a significant drop in the accuracy of the **HSCR** algorithm. This is in stark contrast to other two clustering algorithms for which increasing the window size consistently improves the accuracy. This is because of the inefficiency of **HSCR** algorithm in clustering binary data which suffers from the drawback of getting stuck at local optima that depend on the random values of initial centroids. It turns out that the clusters obtained by this algorithm for window size $w = 3$ are not appropriate at all. This observation also demonstrates the

importance of the hybrid algorithms in exploiting the high quality clusters in the proximity of obtained solutions.

6.5 Convergence behavior

In this part of experiments the convergence of the proposed algorithms will be investigated. Convergence is the amount of time required by a procedure to reach its best possible solution. The less time needed for convergence, the algorithm is faster. However some sorts of algorithms converge sooner than the expected time. This problem is known as premature convergence. The k -means is known as an algorithm with stochastic behavior. There is an assumption that this behavior of k -means has resulted into such late convergence.

Although HS session clustering is to some extent reliant on random number generations, the harmony memory entity prevents excessive stochastic behavior, since the algorithm during its execution, considers harmony memory at each step. Therefore it can be concluded that, HS by reliance on harmony memory **M** is able to reduce the amount of time needed for convergence.

The other proposed variants such as **HKSC**, has showed a better behavior, in terms of convergence, since this sort of hybridization provides the opportunity to benefit from both advantages of HS and k -means, and the procedure continuously alternates between two different amounts of fitness. Convergence behavior of the proposed algorithms is depicted in Fig. 14. It can be observed from the figure that k -means has the worst convergence in terms of number of iterations while other algorithms converge much faster.

6.6 Experiments on clustering quality

In this section we compare the proposed session clustering algorithms to the baseline methods, using two measures of VC and ADSC. The comparisons are made with k -means based session clustering **KSC**, genetic algorithm based clustering **GA**, **WebPUM** and **SUGGEST** algorithms. The parameters of the **GA** method are fine-tuned to their best possible performances. In **GA** we use cut and splice crossover policy in which crossover probability (pc) is 0.6, mutation probability (pm) is 0.1 and genetic pool size is as same as HMS in HS.

In Fig. 15, **HKSC** and **HSC** are performing better than the other variations. **IHKSC** is interleaving between **HSC** and **HKSC** results, which shows the interleaved behaviour of the algorithm. **HKSC** has similar behavior to **HSC** up to iteration 80 and then goes higher and reaches roughly 0.04. Other algorithms like **KSC** and **GA** could not show satisfactory results as much as HS-based variations. Figure 16 shows the average values of the proposed algorithms. According to Fig. 16, **KSC** has the poorest performances while **HSC** is

Table 5 Configuration of the parameters for the conducted experiments obtained by fine-tuning of parameters for different settings

Parameter	Value
# Iterations	200
Harmony memory consideration rate (HMCR)	0.95
Harmony memory Size (HMS)	5
Pitch adjusting rate (PAR)	0.051
# Cluster (K)	40

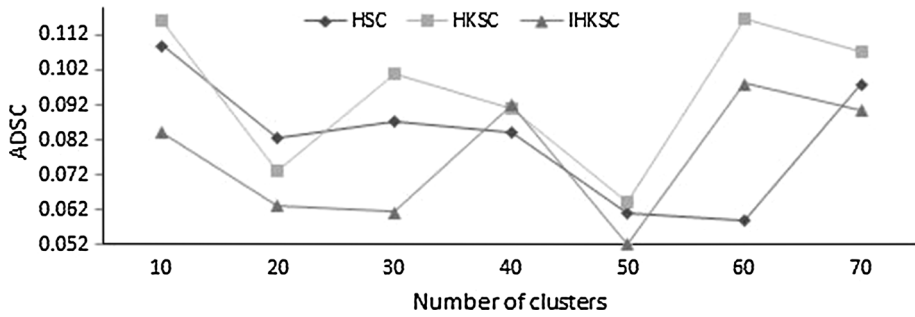


Fig. 8 Experiments on the effect of number of clusters (K) on clustering accuracy measured in terms of ADSC

the best algorithm for clustering the web sessions and generating clusters with low ADSC. The main reason of these significances in the proposed algorithms is on one hand the ability of HS algorithm in performing global search and on the other hand its integration with k -means as a suitable pair for local search that led to improved results.

In Fig. 17 we used VC to make relevant comparisons with other works that consider session clustering in their recommendation process. The performances of **HKSC** variation comparing to the competitors is not satisfactory enough for values $0.5 < \text{MinFreq} < 0.9$. Hence we can conclude that the proposed variations are good enough in producing clusters with high quality in terms of ADSC while not good enough in producing clusters with high rates of VC.

6.7 Experiments on recommendation performance

In this section we present a set of experiments and comparisons to evaluate the performance of recommendation algorithms. In particular, the objective is to test how well the proposed recommendation systems can make predictions for future users, compared to the baselines competitors discussed before.

To make the comparisons justifiable with other works including (AlMurtadha et al. 2011) the results are compared in terms of accuracy, coverage and F-measure. The

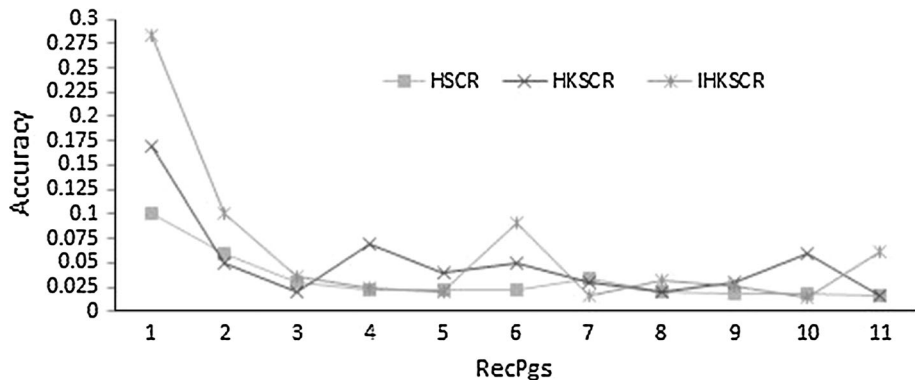


Fig. 9 The accuracy of recommendation algorithms for window size $w = 1$

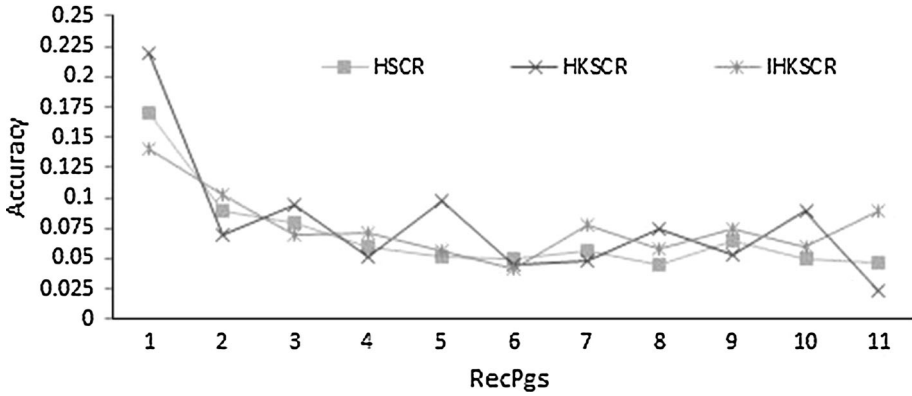


Fig. 10 The accuracy of recommendation algorithms for window size $w = 2$

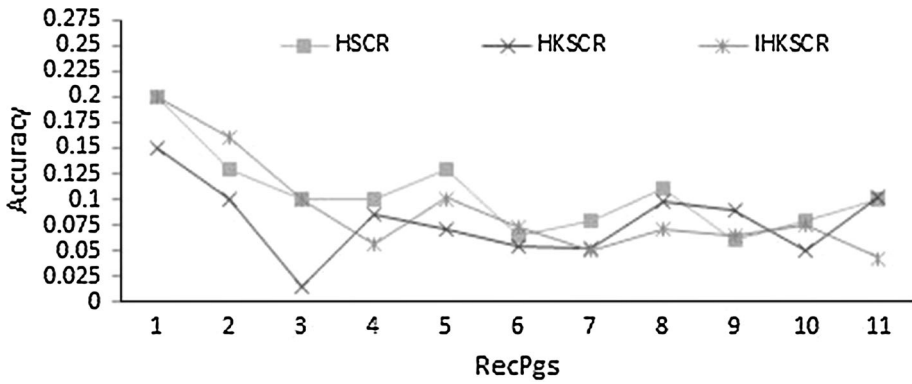


Fig. 11 The accuracy of recommendation algorithms for window size $w = 3$

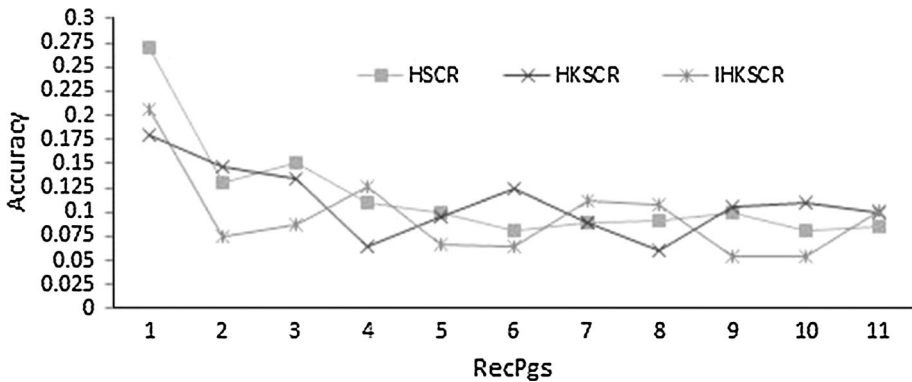


Fig. 12 The accuracy of recommendation algorithms for window size $w = 4$

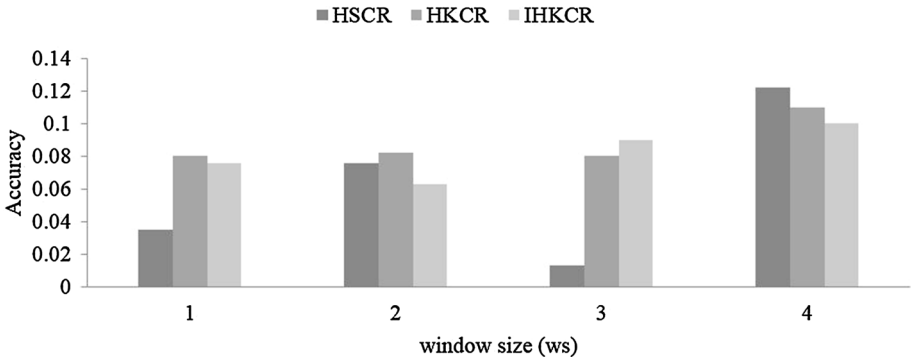


Fig. 13 Average accuracy of algorithms for different values of window size $|w|$

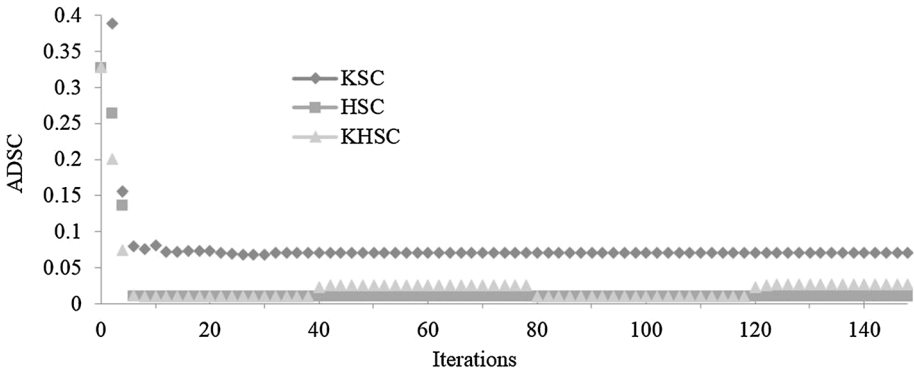


Fig. 14 Convergence of different algorithms measured in terms of ADSC

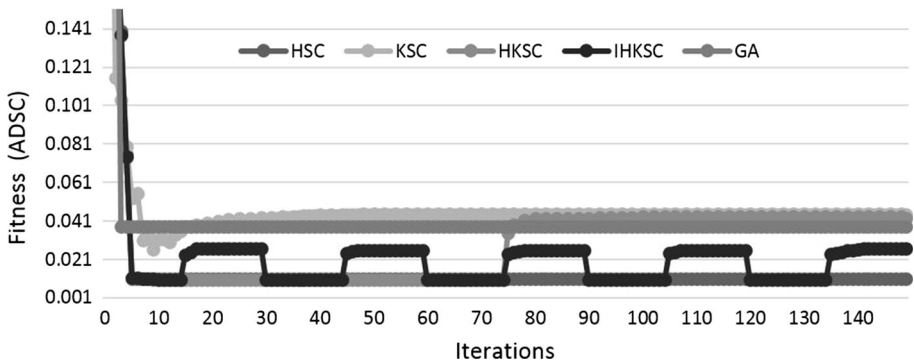


Fig. 15 ADSC comparisons of the proposed methods with genetic and k -means algorithms

recommendation performances are heavily dependent on number of pages that is recommended to the user as a set of recommendation (RecPgs). Also the more pages a system recommends, it is more likely that the users desired Web page to be included in the

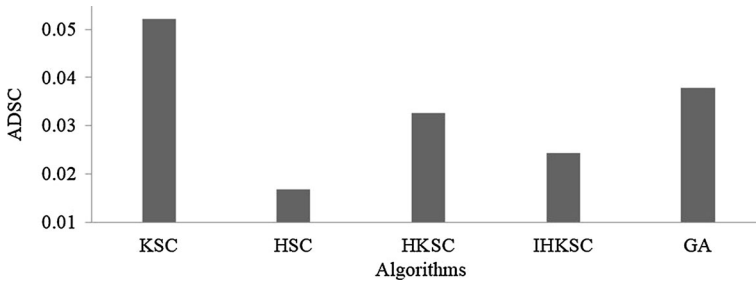


Fig. 16 Average values of the proposed HS-based algorithms compared to GA and *k*-means algorithm measured in terms of ADSC

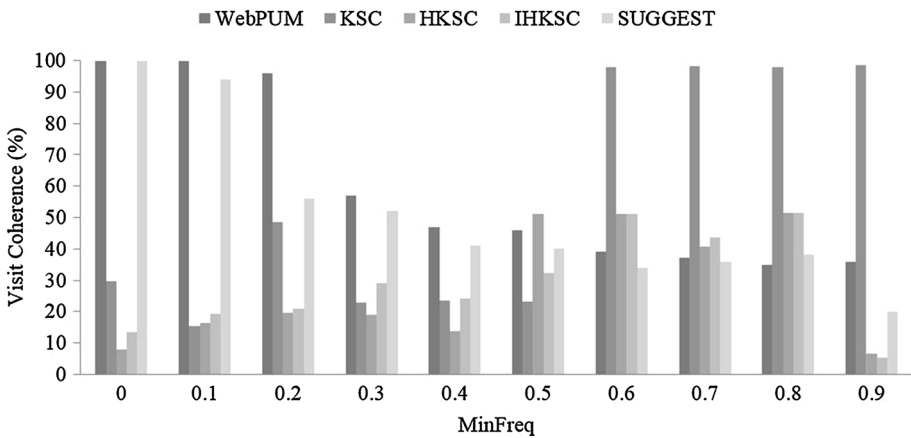


Fig. 17 Comparisons of proposed HS based session clustering algorithms with SUGGEST and WebPUM algorithms measured in terms of visit coherence

recommendation set. Hence the comparisons are made based on the perspective of number of recommended pages.

6.7.1 Optimization based recommender comparisons

In this section we have implemented the GA and *k*-means and replaced them with pure HS algorithm to investigate their effectiveness in making recommendations. At it can be inferred from Fig. 18, the GAR in terms of accuracy could outperform other algorithms, while the performance of KCR is not significant enough comparing to the competitors. In Fig. 19, the coverage of the proposed algorithms is significant comparing to GAR and KCR.

In Fig. 19, HSCR and HKSCR have shown the best performances, and finally Fig. 20 compares the algorithms in terms of F-measure and concludes that the proposed HS based algorithms are effective enough in comparison to GAR and *k*-means variations. In Fig. 20, GAR is inferior to the proposed variations, resulted from its poor coverage rate, while KCR is almost similar to the proposed HS based variations. The comparisons with optimization based algorithms shows that the HS based recommenders can gain satisfactory

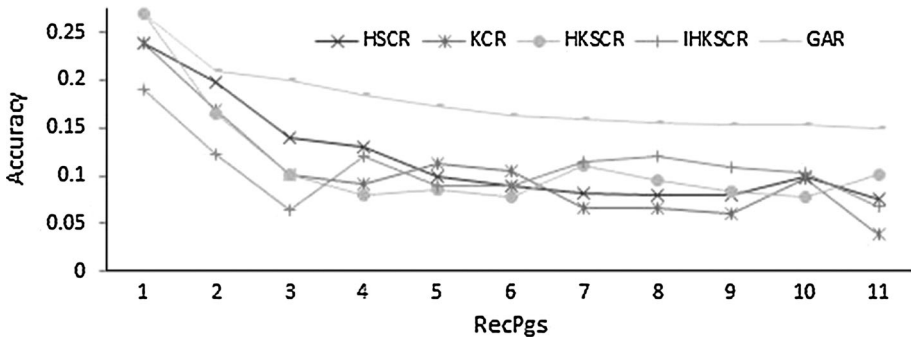


Fig. 18 Comparisons of the proposed algorithms with GA based recommender and *k*-means in terms of accuracy measure

results in term of coverage and F-measure, while their performance comparing to others suffers from low rates of accuracy.

6.7.2 Comparison with graph partitioning based recommenders

In this section we compare our algorithms with graph partitioning algorithms **WebPUM**, **SUGGEST**, **LAGP**, **LCS**. To have understandable comparisons we show the results are shown in the form of percentage. In Figs. 21, 22 and 23 the comparisons are made with **WebPUM** and **SUGGEST**. The **WebPUM** algorithm was outperformed by the proposed variations when $MinFreq < 0.4$ and $MinFreq > 0.8$. Also, **SUGGEST** could show better performances comparing to the competitors when $MinFreq \in (0.2, 0.5)$. In overall the accuracy of the proposed variations is better than the competitors.

In Figs. 22 and 23, the proposed variations are compared with other competitors in terms of coverage and F-measure, respectively. The proposed variations could outperform the competitors in terms of coverage for $MinFreq > 0.1$. Figure 23 concludes that HS based variations are strong enough comparing to graph partitioning variations thanks to their high rates of coverage.

In Figs. 24, 25 and 26, we compare the proposed algorithms with **LCS** recommender as a graph partitioning algorithm. In Fig. 24 the accuracy for different values of $MinFreq$ is varying. Even though the proposed variations could outperform **LCS** for most of the $MinFreq$ values, but in some values of $MinFreq$, **LCS** is superior to HS based algorithms. Furthermore in Figs. 25 and 26, the proposed algorithms are superior over the competitors in terms of coverage and F-measures. Consequently the proposed HS algorithms are superior to **LCS** as a graph based recommender algorithm.

In Figs. 27 and 28 we compare the proposed variations with **LAGP**. Only two measures of accuracy and coverage are used to have justifiable comparisons. **LAGP** shows good results for accuracy while poor results in coverage, comparing to HS based variations. Similar to optimization based comparisons, graph based algorithms showing good results in terms of accuracy while heavily suffer from low rates of coverage and F-measure, comparing to the competitors.

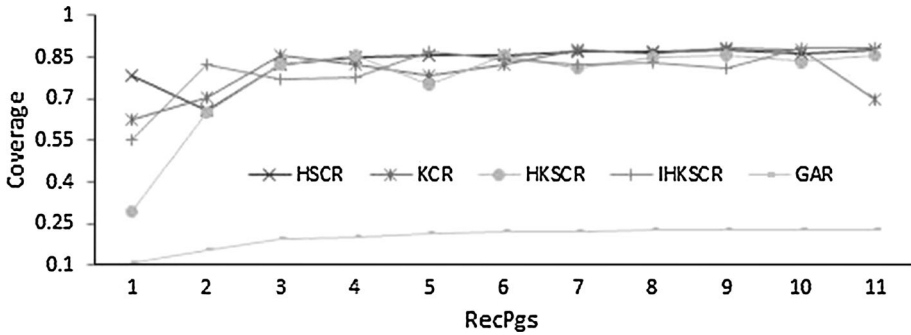


Fig. 19 Comparisons of the proposed algorithms with GA based recommender and *k*-means in terms of coverage measure

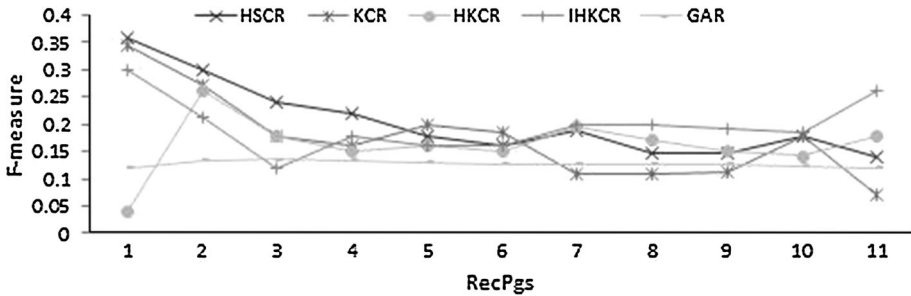


Fig. 20 Comparisons of the proposed algorithms with GA based recommender and *k*-means in terms of F-measure

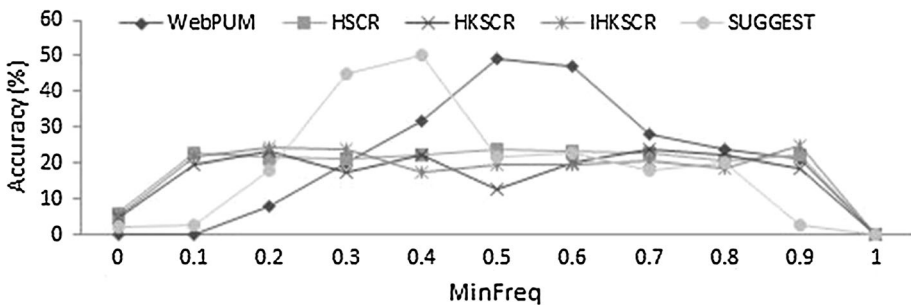


Fig. 21 Comparisons of the proposed algorithm with graph partitioning algorithms using accuracy measure

6.8 Comparison of the proposed algorithms with learning based recommenders

In this section we compare the proposed algorithms with the machine learning based Web page recommendation algorithms. In order to have fair comparisons with the competitors, we have only used two measures of accuracy and coverage. In Figs. 29 and 30 to have

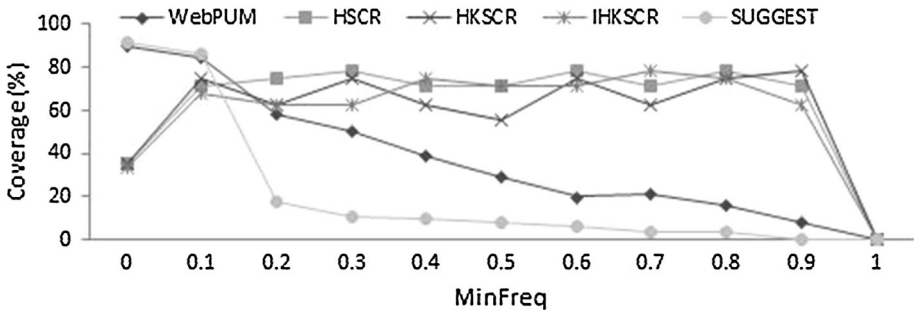


Fig. 22 Comparisons of the proposed algorithms with graph partitioning based algorithms in terms of coverage measure

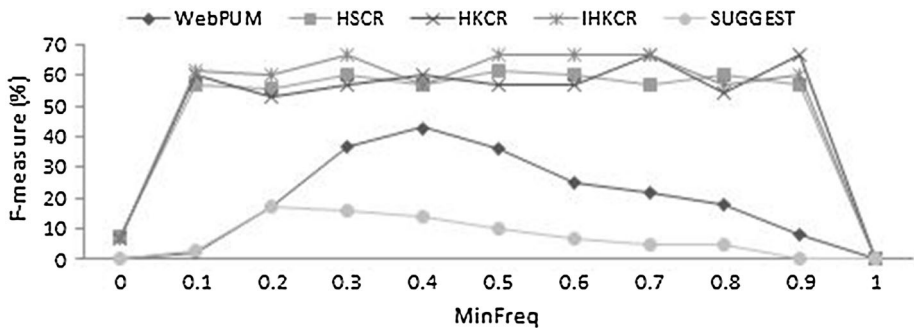


Fig. 23 Comparisons of the proposed algorithms with graph partitioning based algorithms in terms of F-measure

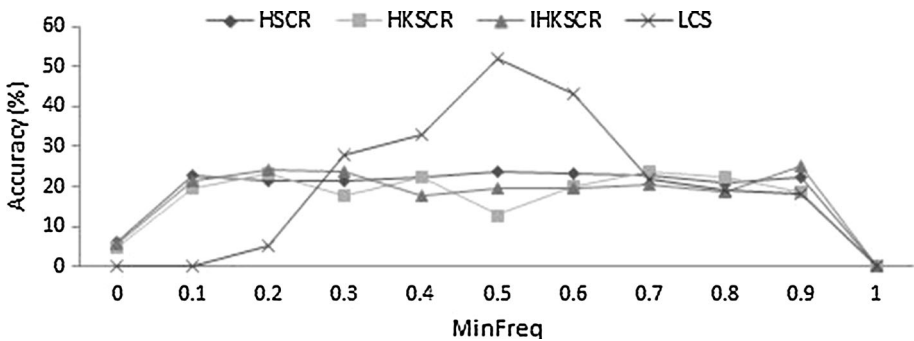


Fig. 24 Comparisons of the proposed algorithms with LCS in terms of accuracy

justifiable comparisons we have shown the results in the range (0–1). In Fig. 29 the superiority of the **DLA** algorithm, the **WARM**, and the **DLA+WARM** algorithms are significant in terms of accuracy, comparing to HS based variations. In contrast in coverage comparisons the proposed algorithms are performing better.

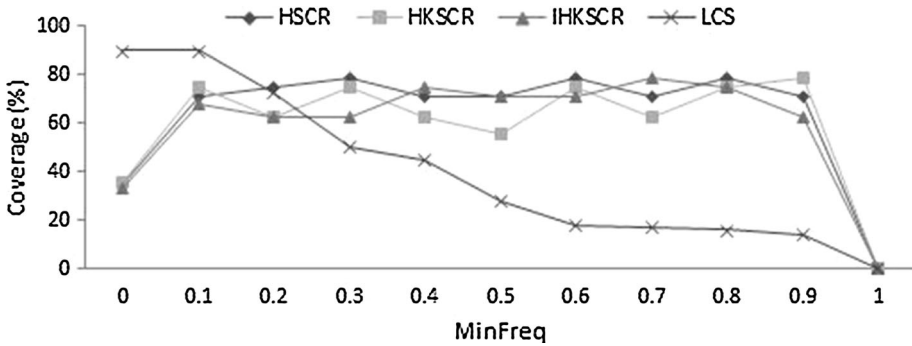


Fig. 25 Comparisons of the proposed algorithms with LCS in terms of coverage

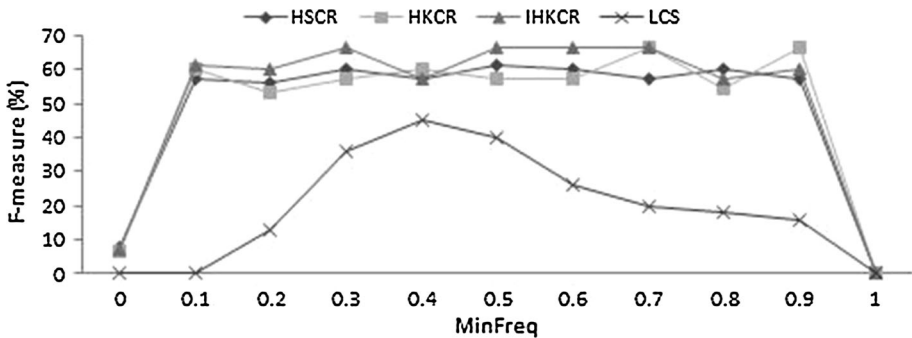


Fig. 26 Comparisons of the proposed algorithms with LCS in terms of F-measure

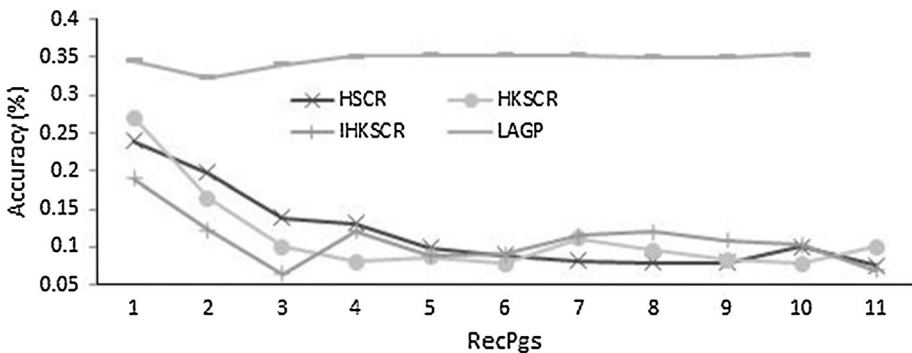


Fig. 27 Comparisons of the proposed algorithms with LAGP in terms of accuracy

Also in Fig. 31, the results of comparison with ARM, DCLA are shown based on the relation between accuracy and coverage. In the proposed variations the least coverage value was 65 %. Hence the results of HS are string from 65. Again in this set of comparisons the proposed algorithms have outperformed the competitor in terms of coverage while have poor performances in accuracy.

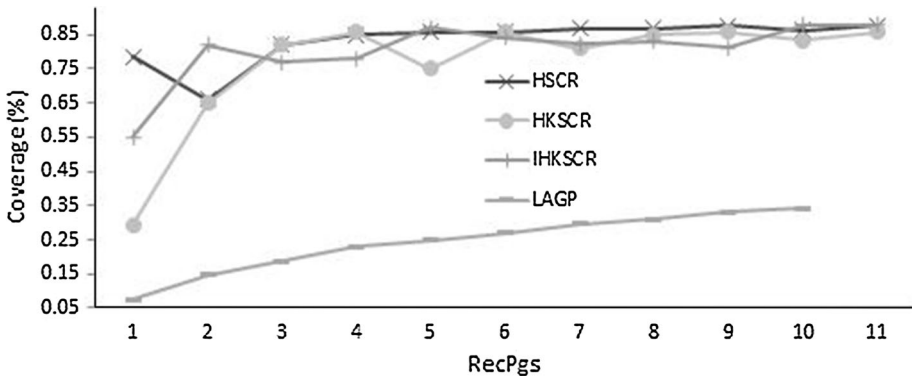


Fig. 28 Comparisons of the proposed algorithms with LAGP in terms of coverage

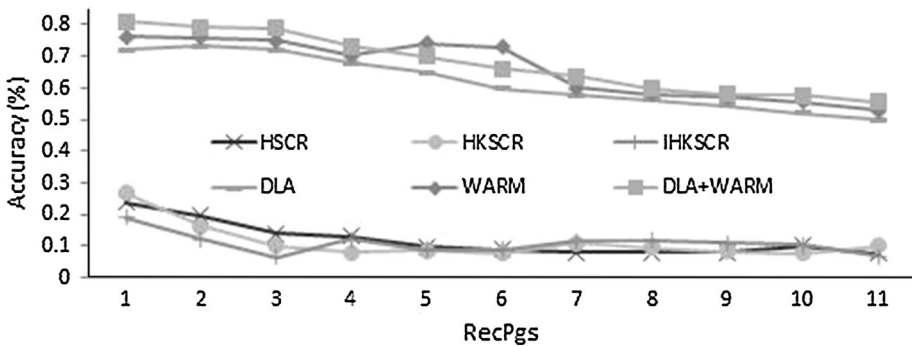


Fig. 29 Comparisons of the proposed algorithms with other learning based algorithms in terms of accuracy

6.9 Other comparisons

In this section we compare our proposed HS-based algorithms with **IPACT** which can be classified neither as a graph-based, optimization or machine learning based algorithm. According to Fig. 32 in accuracy measure **IPACT** is superior to the HS-based variations, resulted from the poor initialization of HS. Even though Fig. 33 indicates that **IPACT** is inferior to HS based variations in coverage measure, but based on the results shown in Fig. 34, F-measure indicates similar performances of **IPACT** and the HS-based session clustering algorithms. In F-measure only the interleave variation (**IHKSCR**) could outperform **IPACT**.

With no exception the comparisons with optimization based, graph based and machine learning based variations have shown to have superior performances over the proposed HS based variations in terms of accuracy, while HS based algorithms are outperforming the competitors in terms of coverage and F-measure.

The poor results of **HSCR** in accuracy can be caused as a consequence of problem modeling. In the proposed algorithms each NHV, has the length equal to the number of sessions, while each component of a NHV will decide that the *i*th session to be assigned to

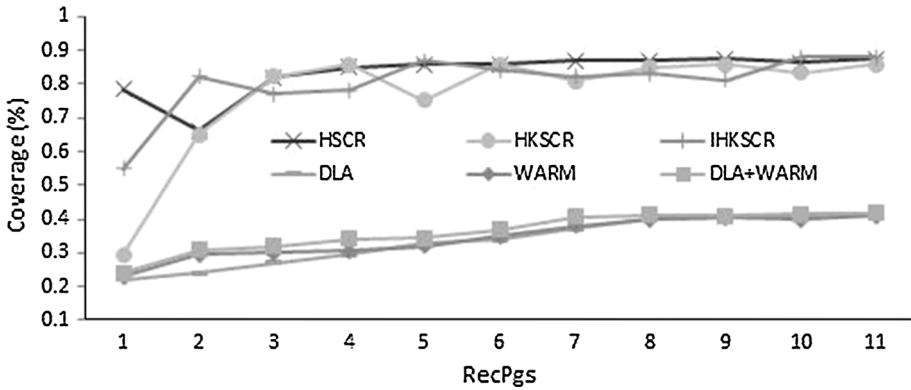


Fig. 30 Comparisons of the proposed algorithms with other learning based algorithms in terms of coverage

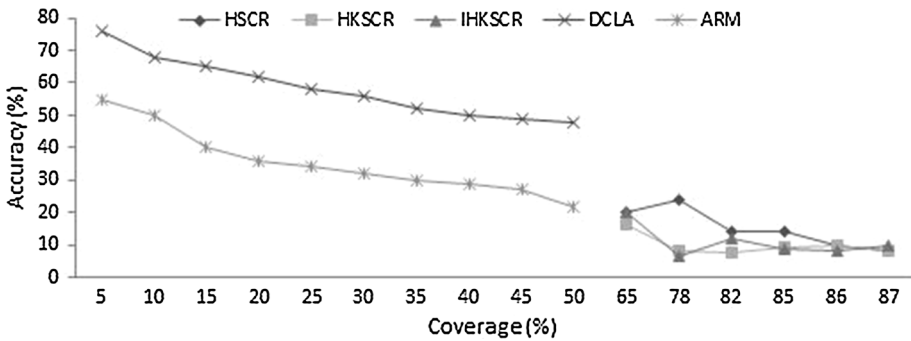


Fig. 31 Comparisons of the proposed algorithms with other learning based algorithms in terms of accuracy and coverage

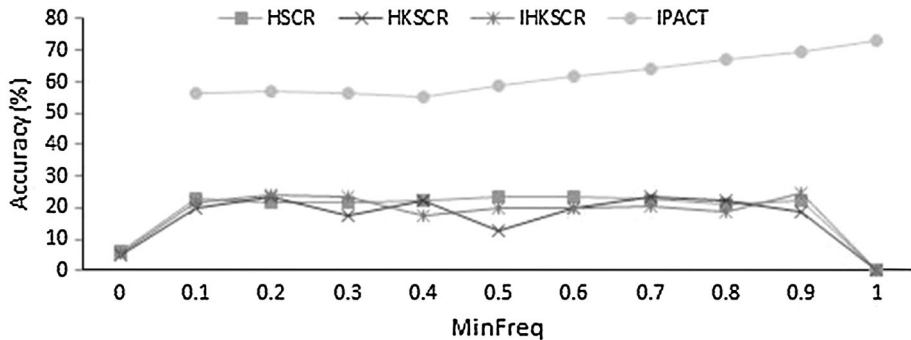


Fig. 32 Comparisons of the proposed algorithms with IPACT in terms of accuracy

the c th cluster. Apparently no heuristic is used (e.g. distances function, semantics, or etc.) to assign sessions to a cluster. Therefore this procedure will lead to low rates of accuracy. Since **HSCR** is the underlying algorithm for other variations, it will affect **HKSCR** and

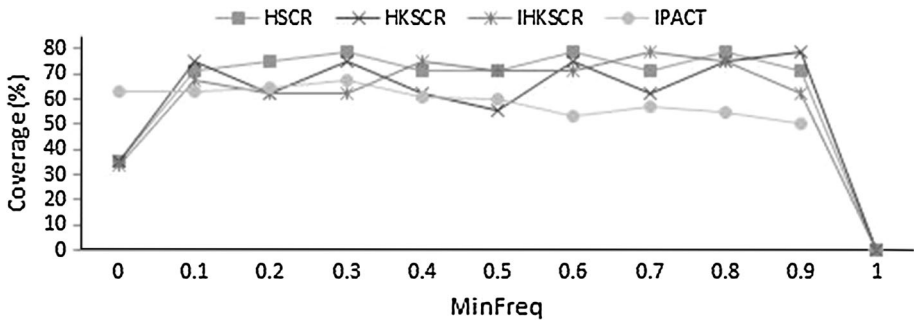


Fig. 33 Comparisons of the proposed algorithms with IPACT in terms of coverage

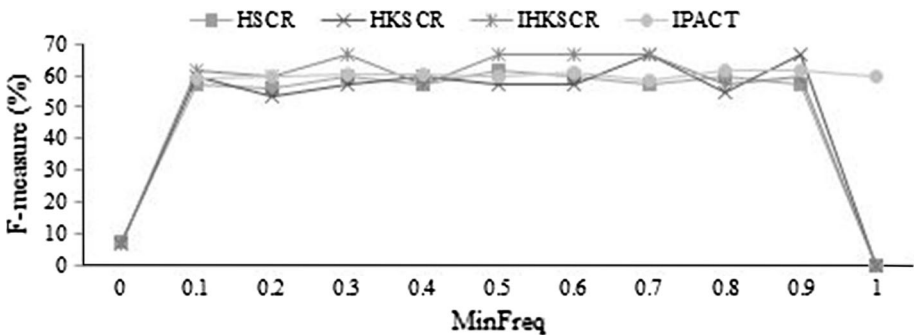


Fig. 34 Comparisons of the proposed algorithms with IPACT in terms of F-measure

IHKSCR correspondingly. In spite of this fact, the proposed variations outperform the competitors as the F-measure results indicate, thanks to high rates of coverage.

6.10 Experiments on re-clustering performance

In this section we conduct some experiments to investigate the effectiveness of the proposed algorithms in handling dynamic page recommendation.

The experiments are carried out where the dynamic parameters are set according to PageFreq = 0.5, window size $w = 4$ and the number of recommended pages $r = 2$, as the previous experiments showed these values are best for these dynamic parameters. We considered different ranges for re-clustering threshold, in the interval of [0.01, 0.05], as shown in Figs. 35, 36 and 37. In these experiments we randomly eliminated 10% of the pages from training sessions and then consider them as newly added pages, while handling test sessions.

The proposed variations showed low rates of accuracy while had remarkable values in two other measures of coverage and F-measure. The reason would be the way that HS has modelled the clustering of sessions. In this model, each column of M or a created NHV represents a session. Then each component is filled with a cluster number. This modeling does not consider any relation or similarity between the data reside in the clusters. Hence in HSC accuracy is low. In KSC the logic is to assign the clusters based on a similarity metric (e.g. Euclidean distance). Hence the results of KSC in terms of accuracy are better than

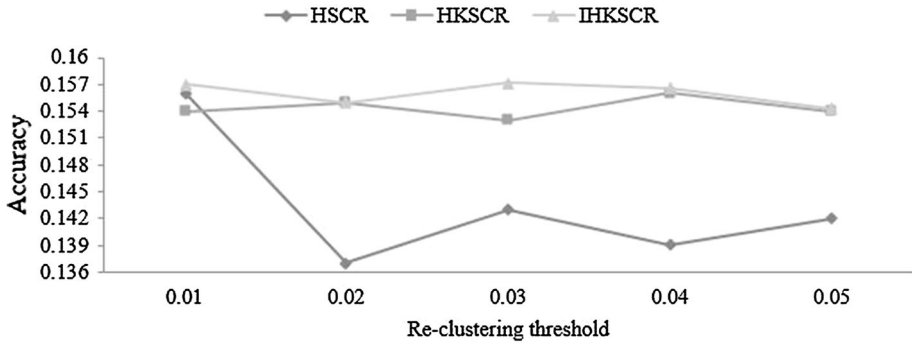


Fig. 35 Accuracy of the proposed algorithms in handling dynamic pages

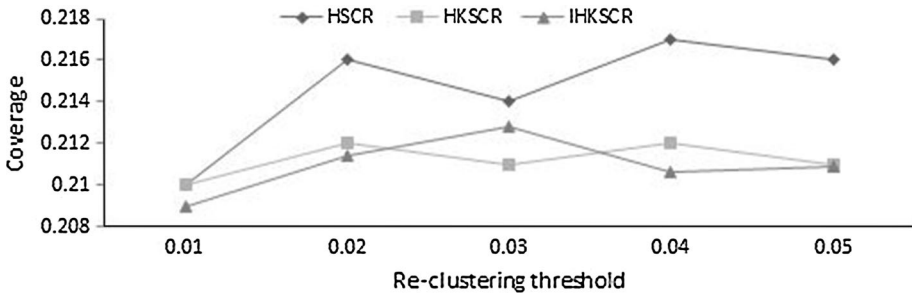


Fig. 36 Coverage of the proposed algorithms in handling dynamic pages

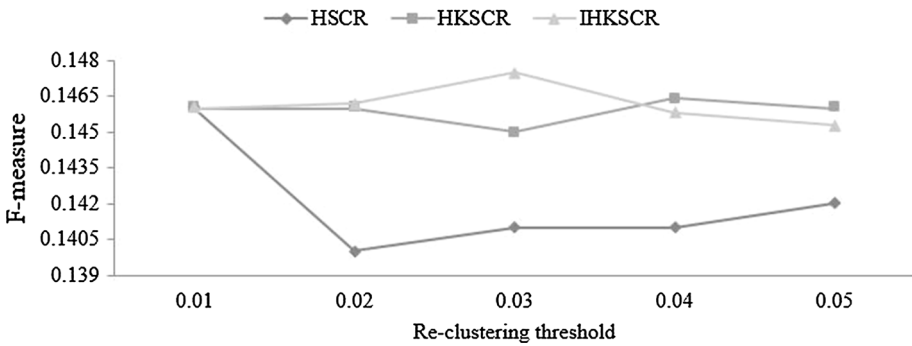


Fig. 37 F-measure of the proposed algorithms in handling dynamic pages

HSC. The integration of **KSC** and **HSC** showed better accuracy comparing to **HSC**, as it utilizes the *k*-means logic in assigning data to their most suitable cluster.

In terms of coverage measure, in all cases by increasing the number of recommended pages (RecPgs), coverage also increases, thanks to modeling of **M**. To see this, assume each NHV with the length equal to the number of clusters. Hence in this case some of the training sessions would not find any cluster to reside in. Hence we will miss some data. But in the proposed modeling in which each column is a session id, the whole data will reside

in one and only on cluster. As **HSC** is the basis of the proposed RS, it will affect negatively or positively the recommendation process. Therefore it is concluded that superiorities in coverage is related to the HS modeling.

Comparisons of the proposed variations in terms of accuracy indicate the superiorities of **HKSCR** and **IHKSCR** variations over **HSCR**, thanks to hybridization process. In spite of poor results of **HSCR** in terms of accuracy, it showed good performances in coverage, as a result of problem modeling, in which each column of **M** corresponds to a session, which increases the coverage value, while its integration with *k*-means will affect the **M** structure and reduces coverage. Finally superiorities of **HSCR** in coverage will help the algorithm to show good results in F-measure comparing to **HKSCR** and **IHKSCR**. In RS, F-measure will decide about the performances of the recommendation algorithm, if it is satisfactory or not, however accuracy plays very important role as it explains the recommendation ability in accurately proposing webpages to the users.

7 Conclusions

In this paper, we have addressed the problem of partitioning a set of binary vectors into a pre-specified number of clusters. We proposed a novel idea to compute the representative of a set of binary vectors and applied the HS optimization method to explore the solution space to find a near-optimal solution. To improve the quality of the basic HS based clustering method, we integrate it with the *k*-means method to gain the advantages of both methods, i.e., the explorative power of HS and fine-tuning capability of *k*-means. We employed the proposed binary clustering method to develop an efficient Web page recommendation algorithm. We have performed extensive experiments on a real dataset to evaluate the quality of the proposed binary clustering algorithm and the efficiency of the RS. The obtained results demonstrated the merits of our recommender algorithm in comparison to other algorithms in terms of F-measure and VC for Web page recommendation.

This work leaves few directions as future work that are worthy of investigation. First, it would be interesting to extend our model by utilizing other sources of side information about Web site such as content of pages and the structure of Web site to handle cold-start and dynamic pages. Also, to measure the similarity of binary session vectors, we simply use the Hamming distance between two binary vectors. So, an interesting direction would be to use other measures such as the Levenshtein distance to compute the similarity of sessions. Leveraging the ontology of Web site to enhance the Levenshtein distance in measuring the similarity of sessions is another interesting direction.

Acknowledgments The authors would like to thank the Associate Editor and anonymous reviewers for their immensely insightful comments and helpful suggestions on the original version of this paper. The authors also would like to acknowledge the assistance of Professor Robin Burke, School of Computing, DePaul University in reviewing early drafts of this manuscript.

References

- Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749.

- AlMurtadha, Y., Sulaiman, M. N., Mustapha, N., & Udzir, N. I. (2011). Ipaact: Improved web page recommendation system using profile aggregation based on clustering of transactions. *American Journal of Applied Sciences*, 8(3), 277.
- Baraglia, R., & Silvestri, F. (2007). Dynamic personalization of web sites without user intervention. *Communications of the ACM*, 50(2), 63–67.
- Billsus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. In *ICML*, vol. 98, (pp. 46–54).
- Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 43–52). Morgan Kaufmann Publishers Inc.
- Chen, G., Wang, F., & Zhang, C. (2009). Collaborative filtering using orthogonal nonnegative matrix tri-factorization. *Information Processing and Management*, 45(3), 368–379.
- Chen, L., Bhowmick, S. S., & Nejd, W. (2009). Cowes: Web user clustering based on evolutionary web sessions. *Data and Knowledge Engineering*, 68(10), 867–885.
- Duraiswamy, K., & Mayil, V. V. (2008). Similarity matrix based session clustering by sequence alignment using dynamic programming. *Computer and Information Science*, 1(3), P66.
- Einaki, M., & Vazirgiannis, M. (2007). Web site personalization based on link analysis and navigational patterns. *ACM Transactions on Internet Technology (TOIT)*, 7(4), 21.
- Forsati, R., Dousdar, H. M., Shamsfard, M., Keikha, A., & Meybodi, M. R. (2013). A fuzzy co-clustering approach for hybrid recommender systems. *International Journal of Hybrid Intelligent Systems*, 10(2), 71–81.
- Forsati, R., Haghghat, A. T., & Mahdavi, M. (2008). Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing. *Computer Communications*, 31(10), 2505–2519.
- Forsati, R., Keikha, A., & Shamsfard, M. (2015). An improved bee colony optimization algorithm with an application to document clustering. *Neurocomputing*, 159, 9–26.
- Forsati, R., & Mahdavi, M. (2010). Web text mining using harmony search. In *Recent advances in harmony search algorithm* (pp. 51–64). Springer.
- Forsati, R., Mahdavi, M., Shamsfard, M., & Meybodi, M. R. (2013). Efficient stochastic algorithms for document clustering. *Information Sciences*, 220, 269–291.
- Forsati, R., & Meybodi, M. R. (2010). Effective page recommendation algorithms based on distributed learning automata and weighted association rules. *Expert Systems with Applications*, 37(2), 1316–1330.
- Forsati, R., & Shamsfard, M. (2014). Novel harmony search-based algorithms for part-of-speech tagging. *Knowledge and Information Systems*, 42(3), 709–736.
- Fu, Y., Sandhu, K., & Shih, M.-Y. (1999). Clustering of web users based on access patterns. In *Proceedings of the 1999 KDD Workshop on Web Mining*. Technical Report. Citeseer.
- Fu, Y., Sandhu, K., & Shih, M.-Y. (2000). A generalization-based approach to clustering of web usage sessions. In *Web Usage Analysis and User Profiling* (pp. 21–38). Springer.
- Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2), 60–68.
- Herlocker, J. L., Konstan, J. A., Borchers, A., Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 230–237). ACM.
- Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1), 89–115.
- Jalali, M., Mustapha, N., Mamat, A., & Sulaiman, M. N. B. (2009). A recommender system for online personalization in the wum applications. In *Proceedings of the World congress on Engineering and Computer Science*, vol. 2, (pp. 20–22).
- Jalali, M., Mustapha, N., Mamat, A., & Sulaiman, N. B. (2008). A new clustering approach based on graph partitioning for navigation patterns mining. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on IEEE* (pp. 1–4).
- Jalali, M., Mustapha, N., Sulaiman, M. N., & Mamat, A. (2010). Webpum: A web-based recommendation system to predict user future movements. *Expert Systems with Applications*, 37(9), 6201–6212.
- Kim, Y. (2007). Weighted order-dependent clustering and visualization of web navigation patterns. *Decision Support Systems*, 43(4), 1630–1645.
- Kohrs, A., & Merialdo, B. (1999). Clustering for collaborative filtering applications. *Computational Intelligence for Modelling, Control and Automation '99: Intelligent Image Processing, Data Analysis and Information Retrieval*, 3, 199.
- Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8), 30–37.

- Krishna, K., & Murty, M. Narasimha. (1999). Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(3), 433–439.
- Kumar, P., Krishna, P. R., & Bapi, R. S. (2007). Rough clustering of sequential data. *Data and Knowledge Engineering*, 63(2), 183–199.
- Kuo, R. J., Liao, J. L., & Tu, C. (2005). Integration of art2 neural network and genetic k-means algorithm for analyzing web browsing paths in electronic commerce. *Decision Support Systems*, 40(2), 355–374.
- Kyung-Yong, J., & Jung-Hyun, L. (2002). Prediction of user preference in recommendation system using associative user clustering and bayesian estimated value. In *AI 2002: Advances in Artificial Intelligence* (pp. 284–296). Springer.
- Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194(36), 3902–3933.
- Livnat, A., Papadimitriou, C., Pippenger, N., & Feldman, M. W. (2010). Sex, mixability, and modularity. *Proceedings of the National Academy of Sciences*, 107(4), 1452–1457.
- Madhuri, B. C., Chandulal, A. J., Ramya, K., & Phanindra, M. (2011). Analysis of users' web navigation behavior using grpa with variable length markov chains. *International Journal of Data Mining & Knowledge Management Process*, 1(2), 1001–1021.
- Mahdavi, M., Chehreghani, M. H., Abolhassani, H., & Forsati, R. (2008). Novel meta-heuristic algorithms for clustering web documents. *Applied Mathematics and Computation*, 201(1), 441–451.
- Mehr, S. M., Taran, M., Hashemi, A. B., & Meybodi, M. R. (2011). A new recommendation algorithm using distributed learning automata and graph partitioning. In *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on IEEE* (pp. 351–357).
- Mirkhani, M., Forsati, R., Shahri, A. M., & Moayedikia, A. (2013). A novel efficient algorithm for mobile robot localization. *Robotics and Autonomous Systems*, 61(9), 920–931.
- Mobasher, B., Dai, H., Luo, T., & Nakagawa, M. (2001). Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd International Workshop on Web Information and Data Management* (pp. 9–15). ACM.
- Ntoutsis, I., Stefanidis, K., Norvag, K., & Kriegel, H.-P. (2012). greco: A group recommendation system based on user clustering. In *Database systems for advanced applications* (pp. 299–303). Springer.
- Park, S., Suresh, N. C., & Jeong, B.-K. (2008). Sequence-based clustering for web usage mining: A new experimental framework and ann-enhanced k-means algorithm. *Data and Knowledge Engineering*, 65(3), 512–543.
- Salakhutdinov, R., & Mnih, A. (2008). Probabilistic matrix factorization. *Advances in Neural Information Processing Systems*, 20, 1257–1264.
- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic C* (pp. 158–167). ACM.
- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Application of dimensionality reduction in recommender system—a case study. Technical report. DTIC Document
- Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology*, vol. 1. Citeseer.
- Shinde, S. K., & Kulkarni, U. V. (2008). A new approach for on line recommender system in web usage mining. In *Advanced Computer Theory and Engineering, 2008. ICACTE'08. International Conference on IEEE* (pp. 973–977).
- Talabeigi, M., Forsati, R., & Meybodi, M. R. (2010a). A dynamic web recommender system based on cellular learning automata. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on IEEE*, vol. 7, (pp. V7–755).
- Talabeigi, M., Forsati, R., & Meybodi, M. R. (2010b). A hybrid web recommender system based on cellular learning automata. In *Granular Computing (GrC), 2010 IEEE International Conference on IEEE* (pp. 453–458).
- Wan, M., Jönsson, A., Wang, C., Li, L., & Yang, Y. (2012). Web user clustering and web prefetching using random indexing with weight functions. *Knowledge and Information Systems*, 33(1), 89–115.
- Wang, F., Ma, S., Yang, L., & Li, T. (2006). Recommendation on item graphs. In *Data Mining, 2006. ICDM'06. Sixth International Conference on IEEE* (pp. 1119–1123).
- Wang, J., Robertson, S., de Vries, A. P., & Reinders, M. J. T. (2008). Probabilistic relevance ranking for collaborative filtering. *Information Retrieval*, 11(6), 477–497.
- Wang, Y.-T., & Lee, A. J. T. (2011). Mining web navigation patterns with a path traversal graph. *Expert Systems with Applications*, 38(6), 7112–7122.

- Wei, Y. Z., Moreau, L., & Jennings, N. R. (2005). Learning users' interests by quality classification in market-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 17(12), 1678–1688.
- Yang, S-H., Long, B., Smola, A., Sadagopan, N., Zheng, Z., & Zha, H. (2011). Like like alike: Joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web* (pp. 537–546). ACM.
- Ypma, A., Ypma, E., & Heskes, T. (2002). *Categorization of web pages and user clustering with mixtures of hidden markov models*. Technical Report.
- Zhang, S., Wang, W., Ford, J., & Makedon, F. (2006). Learning from incomplete ratings using non-negative matrix factorization. In *SDM*, vol. 6, (pp. 548–552). SIAM.
- Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. In *ACM SIGMOD Record*, vol. 25, (pp. 103–114). ACM.
- Zhang, Y., & Koren, J. (2007). Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, (pp 47–54). ACM.
- Zhou, D., Zhu, S., Yu, K., Song, X., Tseng, B. L., Zha, H., & Giles, C. L. (2008). Learning multiple graphs for document recommendations. In *Proceedings of the 17th international conference on World Wide Web* (pp. 141–150). ACM.