Efficient Channel Selection Using Hierarchical Clustering

Hyoungshick Kim
University of British Columbia
hyoung@ece.ubc.ca

Jon Crowcroft
University of Cambridge
jon.crowcroft@cl.cam.ac.uk

Fernando M.V. Ramos
University of Cambridge
University of Lisbon
fernando.ramos@cl.cam.ac.uk

***Abstract*—Increases in the number of TV channels requires users to spend more time to select their preferred channels since the user interaction for browsing is practically limited to the conventional remote control with a two-way scrolling button. We formally define the problem to construct the optimal channel ordering which minimizes the seek distance in selecting channels and show this problem is *NP-hard*. In addition, we present a reasonable heuristic to solve this problem. The proposed method constructs an efficient channel ordering by applying a hierarchical clustering algorithm based on the frequencies of switching events between channels. We demonstrate the feasibility of this method by applying a number of well-known hierarchical clustering algorithms and evaluating the number of user inputs required for selecting channels. Our experimental results show that the proposed method significantly decreases the number of user inputs compared with the conventional methods.**

***Keywords*-Channel Selection, Channel Clustering, User Interaction**

I. Introduction

According to a recent press release by the European Commission [10], 1,033 TV channels have already been established in the UK alone and more than 245 new channels were launched in the course of 2009 in Europe. With the ever increasing number of available channels (and programs) for TV, either through terrestrial broadcasts, cable, satellite or IPTV systems, selecting which channel to watch next is becoming a tedious and time consuming task for the user. Moreover, with the Internet becoming an increasingly important mechanism for delivering media content, the number of channels offered will expectedly increase even further — for example, a number of video sharing services, such as YouTube, have started to provide interfaces to browse their content for TVs (see Fig. 1). Thus, the problem of efficiently browsing channels is becoming a major issue for TV services.

The most common way to browse TV channels is to search linearly until finding something interesting from bottom to top (or vice versa) [24] even if the intermediate channels are not relevant to a user's personal interest. Unlike in a PC environment, the user interface to browsing is generally limited to the conventional remote control with a two-way scrolling button. Thus, to efficiently navigate through the myriad of channels available today it is important to consider this variable. Another difficulty is that in practice

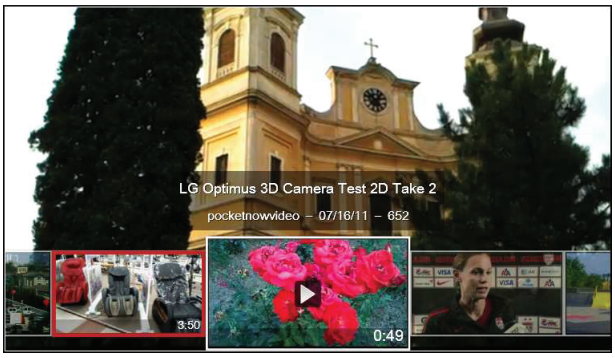


Figure 1. YouTube Leanback. This layout is designed for TV viewing. Playback and browsing can be controlled by a small number of buttons (e.g. a two-way scrolling button) in a remote control.

the browsing information of only a small number of channels can be displayed on the TV screen at any one time. This is mainly due to two factors: the human ability constraints in processing visual information and the limited resolution of TV screens. In addition, the extension of IP connectivity into the realm of mobile and wireless domains where typical devices have small screen sizes makes matters even more complicated in these scenarios.

Our aim in this paper is to develop an efficient navigation method assuming a two-way scrolling button to select a specific channel. The objective is to minimize the *seek distance*, which is defined as the number of user inputs (i.e. the number of times the user presses the `UP` and `DOWN` buttons) to select the requested channel. To this end we propose a new channel ordering construction method for two different types of structures: *Linear* and *Circular*. In a linear structure, the channels are sequentially enumerated from the first channel to the last channel. In a circular structure the first channel is considered as the one next to the last channel of a sequence. Currently, services such as YouTube provide a linear structure of channels, while a circular structure is more common in a TV setting. The key insight behind our proposal is the observation of how users switch between channels (by simply counting the number of switching events between all pairs of channels) and then the rearrangement of channel ordering with the goal of minimizing the seek distance. For example, if a user frequently switches from the "CBeebies" channel to

978-1-4673-1239-4/12$31.00 ©2012 IEEE

the "BBC NEWS" channel, it is desirable to have these two channels closely positioned.

The main contributions of this paper can be summarized in the following points:

1) We formally define the *Minimum User Interaction Problem* to minimize the seek distance in navigating channels, and show this problem is *NP-hard* (Section III).
2) We propose a method to construct an efficient channel ordering by applying a hierarchical clustering algorithm based on user watching history (Section IV).
3) We demonstrate the feasibility of our method by evaluating its performance with several synthetically-generated data traces that follow the Zipf distribution, which is widely-used for channel popularity in related studies [18], [17].

We tested three different cluster selection schemes, namely, single linkage (or nearest neighbour) [16], *unweighted* pair group method with averaging (UPGMA) [28], and *weighted* pair group method with averaging (WPGMA) [27]. The experimental results (Section V) show that the method based on single linkage significantly decreases the number of user inputs for linear structures while maintaining a comparable performance to the best known algorithm for circular structures.

II. Related work

There have been a number of studies into user behavior in IPTV systems. Cha et al. [7] have analyzed a huge dataset from an operational IPTV provider and discussed channel switching behaviors. In particular, they recommended sorting channels based on their popularity (i.e. watching frequency) to reduce unnecessary channel changes. By also analyzing real data collected from a different nation-wide operational IPTV network, Qiu et al. have modeled the popularity of TV channels [23] and have investigated several intrinsic characteristics of IPTV user behavior [22]. In the latter the authors have also developed a synthetic IPTV workload generator that captures both the probability distribution and the time-dynamics of user activities.

As the number of TV channels increases, providing channel recommendation services is becoming an essential functional component of IPTV. Personalized Electronic Program Guides (EPG) [9] are now usually part of the bundle of services offered, helping users to easily find the programs they want to watch. An overview of personalized EPG systems can be found in [26]. These recommendation systems are usually classified into the following two categories: *content-based recommendation* and *collaborative recommendation*. *Content-based* systems [21], [19] recommend programs that match a user's personal interest. For instance, programs can be sorted depending on the similarity to the user's preferred programs in the past. Alternatively, *collaborative* methods recommend the programs that similar users prefer. Collaborative recommendation filtering is a powerful technique; there is no need of content description or sophisticated analysis for the content itself. However, this approach does not come without drawbacks. It involves a startup cost associated with gathering enough profile information to make accurate user-similarity measurements. There is also a latency problem, which means that the recommendation of new programs is delayed until these programs have been positively rated by many users. To make matters worse, user rates or feedbacks are often unavailable due to privacy concerns and may be unreliable due to their laziness and carelessness [4]. In order to complement the advantages and disadvantages of these two recommendation approaches, several hybrid systems were also proposed [3], [30], [5]. Recently, several frameworks were presented incorporating social rating data into program content information to aid users in program selection [13], [20]. In particular, the growing popularity of online social networks has encouraged new research in developing practical collaborative recommendation systems by effectively collecting users' preference on contents through their social activities such as rates and comments [15], [6], [12].

More directly related to the study presented here, Lee et al. [18] presented a channel ordering method called *popularity-interleaved* ordering which is suitable for systems employing a circular structure of channels. In their proposed algorithm, the most popular channel is fixed in the center, channels with odd rankings are positioned on its left side, and channels with even rankings on its right side, according to their popularity. This generates an efficient channel list by placing popular channels at adjacent positions for a circular structure. Lee and Lee [17] proposed a different channel navigation method. Instead of a sequence of channels, they used a binary tree where the most selected channel is the root of the *tree*, the second and the third most selected channels are its children, and so forth. In tree view, however, end-user education is additionally needed to memorize the entire path from the root node to the terminal demand node (i.e. demand channel) to select it properly. It is worth mentioning that in [18] and [17] the performance of the proposed algorithms may be overestimated since the same training and testing sets were used in the experiments. Our work is an extension to these approaches, with a focus on the switching relationship between channels. Furthermore, we also formulate the optimal channel ordering problem and show this problem is *NP-hard*.

III. Minimum user interaction problem

To minimize the number of user inputs to select a particular channel with a two-way scrolling button, instead of using a fixed channel ordering defined by a service provider, in this paper we present a method to construct a user's personal channel ordering based on the user's watching

pattern. Here the efficiency of a channel ordering algorithm can be measured as the seek distance (i.e. the number of times the user presses the UP and DOWN buttons) to select a sequence of channels. In this section we show that it is not trivial to construct an optimized channel ordering that minimizes the seek distance even if the user's watching pattern is completely predictable.

We need to define this optimization problem formally to prove its intractability. We name this the *Minimum User Interaction Problem* (MinUI). Given a sequence of m channels a user will watch in the future, $F = (f_1, f_2, \cdots, f_m)$, where $f_i \in [1 \cdots n]$ and n is the total number of channels, the goal of MinUI is to find a channel ordering $\pi : [1 \cdots n] \rightarrow [1 \cdots n]$ that minimizes

$$\texttt{MinUI}(F) = \sum_{i=1}^{m-1} |\pi(f_i) - \pi(f_{i+1})|. \quad (1)$$

$$= \sum_{\substack{i,j \in [1 \cdots n] \\ i<j}} c_{ij} \cdot |\pi(i) - \pi(j)|. \quad (2)$$

Here, c_{ij} is the frequency count of a pair (i,j) when ($i = f_k$ and $j = f_{k+1}$) or ($i = f_{k+1}$ and $j = f_k$) for $1 \leq k < m$.

We show that MinUI is NP-hard by using a reduction from Minimum Linear Arrangement (MinLA) [25], which is a well-known *NP-hard* problem. Given a weighted graph $G = (V, E)$ with $V = [1 \cdots n]$, the goal of MinLA is to find an ordering π of the graph nodes to minimize

$$\texttt{MinLA}(G) = \sum_{(u,v) \in E} w_{uv} \cdot |\pi(u) - \pi(v)|. \quad (3)$$

Here, w_{uv} is the non-negative weight of the edge (u, v) between nodes u and v (if $(u, v) \notin E$ then $w_{uv} = 0$).

Theorem 3.1: There exists a polynomial transformation from MinLA to MinUI.

Proof:

Let us consider an arbitrary graph $G = (V, E)$ given as an input of MinLA. The reduction takes a graph $G = (V, E)$ with $V = [1 \cdots n]$ and constructs a graph $G' = (V \cup \{\epsilon\}, E \cup \{E_\epsilon\})$ so that G' contains either zero or exactly two of the nodes with odd degrees by adding the new node $\epsilon = n + 1$ and new edges E_ϵ between ϵ and the other nodes properly as follows: Add edges between all nodes of odd degree and ϵ to transform the graph G into an Eulerian graph G' since there are no nodes of odd degree. Since the number of nodes of odd degree is always even in the undirected graph, the degree of ϵ is also even. Thus, we can find an Eular path $P = (v_1, v_2, \cdots, v_l)$ in G'. We set $w_{\epsilon v} = 0$ for $v \in V$ to ignore the terms related to ϵ. This preserves the objective function value in the original graph.

From G', we can directly construct the instance of MinUI with $F = (v_1, v_2, \cdots, v_l)$. We also set $c_{v_i v_j} = 0$ if $v_i = \epsilon$ or $v_j = \epsilon$ as above. In this case, (2) and (3) are equivalent since $w_{xy} = c_{xy} = 0$ if $x = \epsilon$ or $y = \epsilon$.

So, since ϵ does not give effect to the computation of the answer, a solution π for MinUI with $F = (v_1, v_2, \cdots, v_l)$ is a solution for MinLA with G. ■

IV. Proposed method to construct a personalized channel ordering

Since MinUI is NP-hard, heuristics are needed to solve this problem efficiently. Moreover, since we cannot predict the exact user's watching pattern in the future, F, our channel ordering algorithm is based on the observation of the past (within a time window) under the assumption that the future pattern is highly correlated with the past pattern. The main idea of the algorithm is to position the pairs of channels users switch between very frequently close to each other. For example, if a user switches frequently from the "CBeebies" channel to the "BBC NEWS" channel, these will be closely positioned.

Given a sequence of the m channels a user watched in the past, $H = (h_1, h_2, \cdots, h_m)$, where $h_i \in [1 \cdots n]$, we construct a channel ordering π_H to minimize the seek distance for future events by taking the following steps (see also Fig. 2):

1) Construct a weighted graph G_H (called *watching graph*) with a sequence of channels, $H = (h_1, h_2, \cdots, h_m)$ where $h_i \in [1 \cdots n]$, whose nodes are the set of all possible channels, $\{1, 2, \cdots, n\}$, and an edge (u, v) is added if there was at least one channel switching event from the channel u to the channel v. The weight of an edge (u, v) is assigned as to be inversely proportional to the *number of switching events* between u and v or v and u (i.e., the graph is undirected). A small weight thus represents two channels that should be close to each other. Otherwise, the number of channels minus one $(n-1)$ is assigned to the weight of the edge (u, v). The resulting running time is $O(m)$.
2) From the constructed weighted graph G_H, find a binary tree T_H (a dendrogram) whose leaves are the channels and whose internal nodes represent nested clusters such that the (Euclidean) distance measure is minimized within clusters and maximized between clusters using an agglomerative hierarchical clustering algorithm [14]. The hierarchical clustering algorithms (single linkage, UPGMA and WPGMA) used in our experiments run in $O(n^2)$ time and $O(n^2)$ space [29].
3) Generate an efficient channel ordering π_H by finding the optimal linear leaf ordering from the dendrogram T_H. We use the algorithm presented in [2] to find the optimal linear leaf ordering. In most TV settings the list of channels is circular (the channel next to the last is the first). For such circular structures we simply connect both ends of π_H to be adjacent to each other.

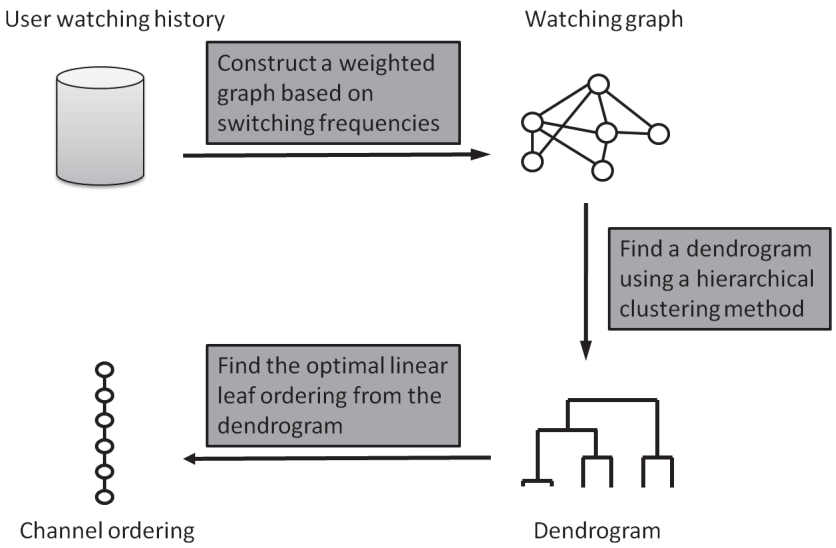


Figure 2. An overview of the proposed method. We construct a linear channel ordering from a user's watching history by using a hierarchical clustering method.

This step runs in $O(n^4)$ time. The total running time of the whole algorithm is therefore $O(n^4 + m)$.

As explained above, for efficient grouping of the channels users switch between very frequently, we opt for an agglomerative hierarchical clustering algorithm. This is because agglomerative hierarchical clustering provides a *nested sequence of clusters* with a single and all-inclusive cluster at the top and single-point clusters at the bottom, unlike partition-based clustering techniques such as k-means [14]. This allows us to compute an efficient channel ordering based on cluster relationships to minimize the sum of distances between all pairs of channels using the optimal leaf ordering algorithm [2].

A. Applying agglomerative hierarchical classification methods

The key parameter in agglomerative hierarchical clustering algorithms is the clustering criterion function used to determine the pairs of clusters to be merged at each step. In most agglomerative algorithms, this is accomplished by selecting the most similar (or closest) pair of clusters. Many cluster selection schemes have been developed for computing the similarity between clusters [16], [14], [27], [11]. They mainly differ in how they update the similarity between existing and merged clusters. In our study we evaluated the following cluster selection schemes: single linkage, UPGMA and WPGMA.

The single linkage scheme [16] measures the distance of two clusters by computing the shortest distance between the channels from each cluster. That is, the distance between two clusters C_x and C_y is given by

$$d_s(C_x, C_y) = \min_{\substack{ch_i \in C_x \\ ch_j \in C_y}} \{\Delta(ch_i, ch_j)\}$$

where ch represent a channel and $\Delta(ch_i, ch_j)$ is the minimum user interaction number to switch the watching channel from ch_i to ch_j or the reverse.

The main disadvantage of the single linkage scheme is that while two clusters may be linked by this technique on the basis of a single bond, many of the channels of the two clusters may be quite far in terms of similarity. This drawback led us to consider the other schemes, namely UPGMA and WPGMA.

The UPGMA scheme [28] measures the distance of two clusters by computing the average of the pairwise distance of the channels from each cluster as follows:

$$d_u(C_x, C_y) = \frac{1}{n_x n_y} \sum_{\substack{ch_i \in C_x \\ ch_j \in C_y}} \Delta(ch_i, ch_j)$$

where n_x and n_y represent the sizes of the clusters C_x and C_y, respectively.

The WPGMA scheme [27] uses a recursive definition for the distance between two clusters. If cluster C_x was created by combining clusters C_{x_1} and C_{x_2}, the distance between C_x and C_y is defined as the average of the distance between C_{x_1} and C_y and the distance between C_{x_2} and C_y as follows:

$$d_w(C_x, C_y) = \frac{(n_{x_1} \cdot d_w(C_{x_1}, C_y) + n_{x_2} \cdot d_w(C_{x_2}, c_y))}{n_{x_1} + n_{x_2}}$$

where C_{x_1} and C_{x_2} are clusters to be merged, and C_x is the cluster such that $C_x = C_{x_1} \cup C_{x_2}$. n_{x_1} and n_{x_2} represent the numbers of the clusters C_{x_1} and C_{x_2}, respectively.

In this paper we use `Single`, `Average` and `Weighted` to denote the single linkage, UPGMA and WPGMA schemes, respectively.

B. An example

To give an intuition for the proposed channel ordering method, we consider an example which consists of 8 channels. Given a sequence of channels, $H = (4, 6, 7, 6, 8, 4, 8, 4, 5, 7, 5, 7, 2, 3, 2, 7, 8, 1)$, we construct a weighted graph G_H as shown in Fig. 3. In this graph, an edge (u, v) is added if there exists a channel switching from u to v or the reverse and the weight of this edge is assigned as to be inversely proportional to the *number of switching events* between u and v. For example, in this graph, the edge (2, 7) has weight 0.5 since there are two channel switches between channels 2 and 7: one from 7 to 2, and other from 2 to 7.

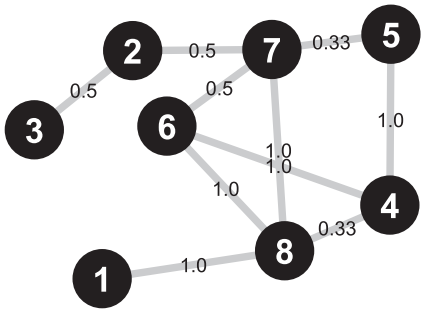


Figure 3. The weighted graph G_H constructed from $H = (4, 6, 7, 6, 8, 4, 8, 4, 5, 7, 5, 7, 2, 3, 2, 7, 8, 1)$. In this graph, each edge is added if there exists a channel switching between them and the edge weight is assigned as to be inversely proportional to the *number of switching events*.

From this weighted graph G_H in Fig. 3, we find a dendrogram T_H whose leaves are the channels using an agglomerative hierarchical clustering algorithm; in order to compare performance, we applied `Single`, `Average` and `Weighted`, respectively. The results are shown in Fig. 4. In this figure, the dendrograms display the linkage distance obtained through cluster analysis. For this example we use a threshold value of (0.7 $\times$ the maximum edge weight of T_H) to represent a cluster which is uniquely coloured in Fig. 4. `Single` and `Average` identify two clusters while `Weighted` identifies three clusters. Finally, from each of these dendrograms we find the optimal linear leaf ordering and use it as a new channel ordering π_H.

Now, given a future sequence of channels, F = (7, 2, 7, 2, 6, 4, 1, 8, 4, 8, 2, 8, 4, 6, 2, 8, 1, 4), we calculate the average seek distances (SD) per channel for linear and circular channel ordering; this can be simply computed as the sum of seek distances to select the channels in F by dividing the size of F minus one ($|F - 1|$). Here we ignore the initial cost (the number of user inputs to select the first channel '7' in F) since the total cost is dominated by the cost of selecting the next channels when the size of F is large. The seek distance to select a channel f_i in F is the distance between the channel f_i and the previous channel f_{i-1}. The experimental results of this example are shown in Table I. For both linear and circular structures, `Single` produces the best results (2.235 for a linear structure and 1.882 for a circular one). In this example, we can see that the channel ordering based on `Single` significantly decreases the seek distance compared with the fixed channels (1–2–3–4–5–6–7–8).

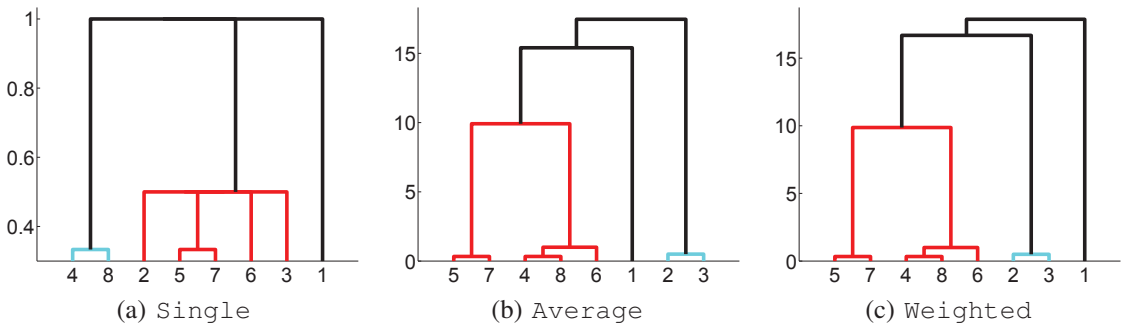
	Channel Ordering	SD for Linear	SD for Circular
`Single`	3–2–6–7–5–4–8–1	2.235	1.882
`Average`	2–3–5–7–6–4–8–1	2.706	2.000
`Weighted`	2–3–5–7–6–4–8–1	2.706	2.000
`Fixed`	1–2–3–4–5–6–7–8	4.529	2.765

Table I
THE RESULTS OF CHANNEL ORDERING FOR $H = (4, 6, 7, 6, 8, 4, 8, 4, 5, 7, 5, 7, 2, 3, 2, 7, 8, 1)$ AND $F = (7, 2, 7, 2, 6, 4, 1, 8, 4, 8, 2, 8, 4, 6, 2, 8, 1, 4)$.

V. PERFORMANCE EVALUATION

To evaluate the efficiency of the proposed method, we synthetically-generated several data traces of channel switching events. These traces follow the Zipf distribution [1] for the popularity of channels, a distribution that was validated by empirical analysis using two different IPTV datasets from real IPTV usage [7], [23], and which is generally used to evaluate the performance of channel ordering methods [18], [17]. In this distribution, the frequency of the rth popular channel is proportional to $r^{-\alpha}$ where α ($0 \leq \alpha \leq 1$) is the Zipf rank exponent parameter that determines the degree of popularity skew. When $\alpha = 0$, all channel popularities are equal, and the distribution gets skewer with α. We tested the efficiency of the proposed method on the several datasets by varying α from 0.2 to 1.0, the size of training samples $m = |H|$ from 1000 to 5000, the size of testing samples $l = |F|$ from 20 to 100, and the number of channels n from 200 to 1000. In order to decrease the bias associated with a set of testing samples, we repeat the test for 200 randomly chosen testing sets and measure the average performance. We compared the performance of our methods (`Single`, `Average` and `Weighted`) with those of the following three methods:

- **random ordering**: A channel ordering is randomly generated. A random channel ordering represents a (fixed) conventional channel ordering. We randomly generated 200 linear channel orderings and computed the average value of them. We use `Random` to denote this scheme.



(a) `Single` (b) `Average` (c) `Weighted`

Figure 4. Dendrograms of the example. Each dendrogram is constructed by using agglomerative hierarchical clustering algorithms with three clustering criterion functions: `Single`, `Average` and `Weighted`. For visualization, edges of each cluster are uniquely coloured.

- **popularity-sorted ordering**: Channels are sorted in descending order of their popularity, since most users are likely to switch more between popular channels. We use `Sorted` to denote this method.
- **popularity-interleaved ordering** [18]: The previous scheme has some limitations for circular structures. This scheme is similar, with the channels sorted in descending order of their popularity, but the structure is different. The position of the most popular channel is fixed, and then channels with odd rankings are placed on its left side, and channels with even rankings on its right side. We use `Interleaved` to denote this scheme.

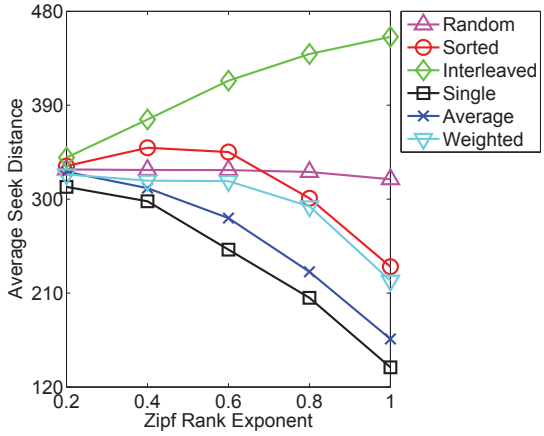


Figure 5. The average seek distance per channel by varying the Zipf rank exponent parameter α for linear structures, with $l = 65$, $m = 5000$ and $n = 1000$.

First of all, we perform an experiment with $l = 65$[1], $m = 5000$ and $n = 1000$ in order to show the effects of the Zipf rank exponent parameter α. The average seek distances per channel via the channel ordering methods are shown in Fig. 5 for the linear structure and in Fig. 6 for the circular structure, respectively.

[1]The average number of switching events per user is around 65 per day [8].

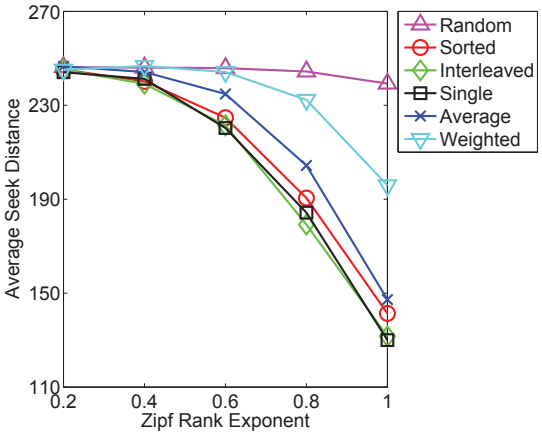


Figure 6. The average seek distance per channel by varying the Zipf rank exponent parameter α for circular structures, with $l = 65$, $m = 5000$ and $n = 1000$.

In linear structures (see Fig. 5), the `Single` ordering methods produced the best results — when $\alpha = 1.0$ `Single` achieved an average seek distance of around 150 while the average seek distance in `Random` was above 300. On the other hand, for circular structures (see Fig. 6), the results were somewhat inconsistent depending on α. The performance of `Single` is almost similar to that of `Interleaved`. Overall, the performance of all ordering methods except `Interleaved` for linear structure improves as α increases; the performance that can be achieved with these ordering methods thus depends on the skewness of channel popularity.

To illustrate the performance of the proposed channel ordering scheme more clearly, we computed the ratio between the results for `Single` channel ordering and the results of the other methods (`Random`, `Sorted`, and `Interleaved`). The results are shown in Table II. For linear structures, `Single` significantly reduces the seek distance on average compared with `Random` and `Sorted`. As shown in Table II, when α = 1.0, `Single` uses only about 43.5% of user inputs of `Random` and 59.0% of user

	Linear Structure		Circular Structure	
	Random	Sorted	Random	Inter.
0.2	0.9491	0.9395	0.9911	0.9956
0.4	0.9084	0.8532	0.9804	1.0087
0.6	0.7676	0.7291	0.8962	0.9951
0.8	0.6302	0.6831	0.7540	1.0284
1.0	0.4353	0.5903	0.5434	0.9879

Table II
RATIO BETWEEN THE RESULTS FOR SINGLE CHANNEL ORDERING AND THE RESULTS OF THE OTHER METHODS.

inputs of Sorted. However, for circular structures Single produced results similar to Interleaved.

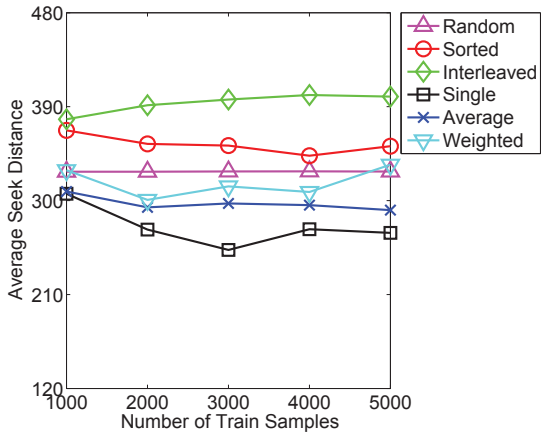


Figure 7. The average seek distance per channel by varying the size of training samples m for linear structures, with $\alpha = 0.55$, $l = 65$ and $n = 1000$.

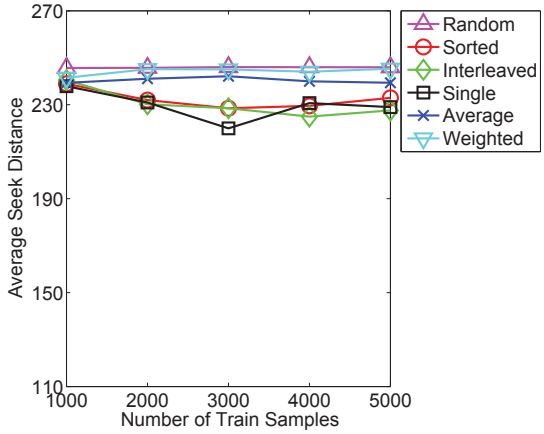


Figure 8. The average seek distance per channel by varying the size of training samples m for circular structures, with $\alpha = 0.55$, $l = 65$ and $n = 1000$.

We now discuss how the performance of the different channel ordering methods may change with the size of the training samples m. To demonstrate this we fix $\alpha = 0.55$[2], $l = 65$ and $n = 1000$ (see Fig. 7 and Fig. 8). The average seek distance of Single overall decreases with m, from $m = 1000$ to 3000. When $m = 3000$, it is clear that Single is better than the other channel ordering methods even for circular structures. For these structures, when the size of the training samples is large (e.g. $m \geq 4000$), Interleaved seems a better option. For linear structures, Single produced the best results overall.

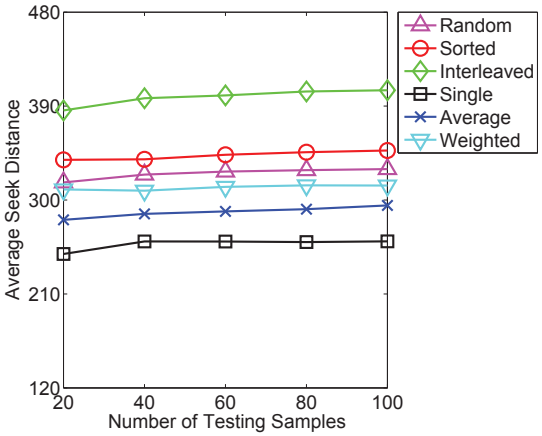


Figure 9. The average seek distance per channel by varying the size of the testing samples l for linear structures, with $\alpha = 0.55$, $m = 5000$ and $n = 1000$.

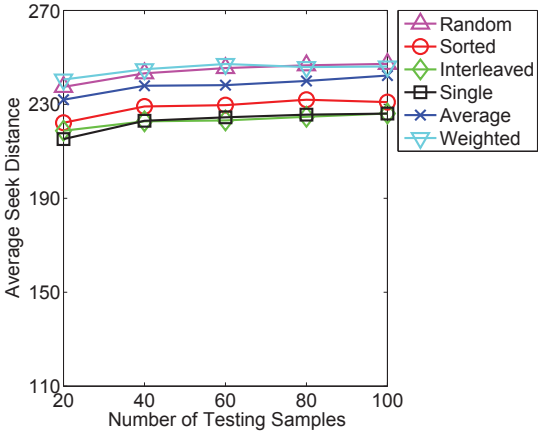


Figure 10. The average seek distance per channel by varying the size of the testing samples l for circular structures, with $\alpha = 0.55$, $m = 5000$ and $n = 1000$.

We now move to the discussion on the performance of these channel ordering methods when the size of the testing samples l changes, by fixing $\alpha = 0.55$, $m = 5000$ and n = 1000 (see Fig. 9 and Fig. 10). The average seek distance

[2]This α value was derived from an empirical study using a set of real IPTV traces [23].

of all methods slightly increases with l similarly. Interestingly, for circular structures `Single` is slightly better than `Interleaved` at $l = 20$. This implies that we recommend using `Single` even for circular structures when l is very small.

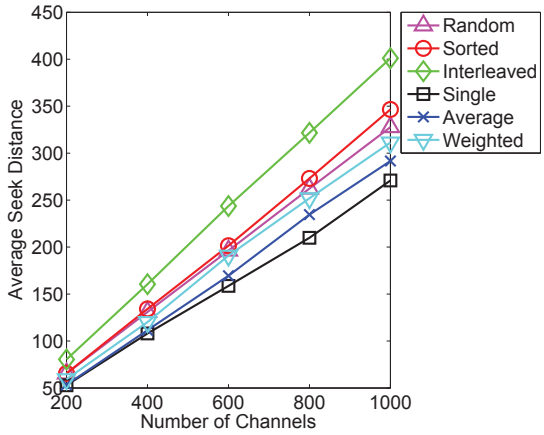


Figure 11. The average seek distance per channel by varying the size of the number of channels n for linear structures, with $\alpha = 0.55$, $m = 5000$ and $l = 65$.

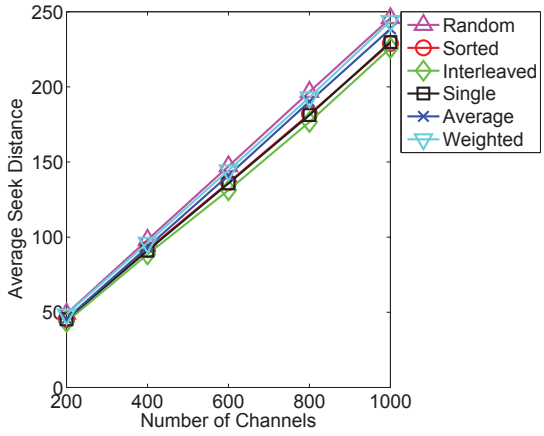


Figure 12. The average seek distance per channel by varying the size of the number of channels n for circular structures, with $\alpha = 0.55$, $m = 5000$ and $l = 65$.

Finally, we now discuss the effects of the number of channels n with $\alpha = 0.55$, $l = 65$ and $m = 5000$ (see Fig. 11 and Fig. 12). From these figures, we can see that the average seek distance generally increases and the gap between the proposed methods and the conventional method `Random` grows with n, although this trend appears to be rather weak for circular structures.

It is important to make it clear that the proposed methods do not come without drawbacks. They involve a possibly resource-intensive training phase, where the software has to analyze the user's watching history in order to learn about the switching relationship between channels. We showed before that the total running time is $O(n^4 + m)$, with n being the number of channels and m the size of the user's watching history. Moreover, users have to become familiar with this newly constructed channel ordering, and learn to select their preferred channels appropriately.

VI. Conclusion

Although the current remote controls provide various features to select TV channels (e.g. restoring the last active channel), previous research has shown that it is very common to switch up or down to the next TV channel linearly, using a two-way scrolling button [24]. Considering this fact, in this paper we propose a novel method to construct a list of TV channels with the objective of minimizing the number of user inputs to effectively navigate them.

We formally define the problem of finding this optimal channel ordering and prove this problem is *NP-hard*. To solve it efficiently, we present a new channel ordering scheme. The main idea is to construct a sequence of channels by applying a hierarchical clustering algorithm in order to bring closer pairs of channels users tend to switch between more frequently. We demonstrate the feasibility of this method with several well-known clustering selection schemes, such as single linkage, UPGMA and WPGMA.

For performance evaluation, we generated synthetic traces of channel switching events based on a statistical distribution which has been validated by previous empirical studies. The experimental results showed that for linear structures the method we propose based on single linkage significantly decreases the number of user inputs on average, while being comparable to the best known algorithm for circular structures [18].

As future work, we plan to develop a working prototype, integrate it with a real TV, and evaluate its usability through a laboratory experiment involving real users. Moreover we do not restrict our attention to the channel ordering problem in TV, but will also extend it to other applications for mobile devices with a horizontal (or vertical) scrolling operation to select content. For example, we plan to apply the proposed method to list-based selection interfaces for media library applications in mobile devices such as the iPhone.

References

[1] L. A. Adamic and B. A. Huberman. Zipf's law and the Internet. *Glottometrics*, 3:143–150, 2002.

[2] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl 1):S22–S29, 2001.

[3] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290–4311, 2010.

[4] A. Basso, M. Milanesio, A. Panisson, and G. Ruffo. On Collaborative Filtering Techniques for Live TV and Radio Discovery and Recommendation. In *EC-Web*, pages 148–159, 2011.

[5] R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[6] P. Cesar, D. C. A. Bulterman, J. Jansen, D. Geerts, H. Knoche, and W. Seager. Fragment, tag, enrich, and send: Enhancing social sharing of video. *ACM Transactions on Multimedia Computing, Communications and Applications*, 5:19:1–19:27, August 2009.

[7] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain. Watching television over an IP network. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 71–84, New York, NY, USA, 2008. ACM.

[8] M. Cha, P. Rodriguez, S. Moon, and J. Crowcroft. On next-generation telco-managed P2P TV architectures. In *Proceedings of the 7th international conference on Peer-to-peer systems*, IPTPS'08, Berkeley, CA, USA, 2008. USENIX Association.

[9] M. Ehrmantraut, T. Härder, H. Wittig, and R. Steinmetz. The personal electronic program guide—towards the pre-selection of individual TV programs. In *CIKM '96: Proceedings of the fifth international conference on Information and knowledge management*, pages 243–250, New York, NY, USA, 1996. ACM.

[10] European Commission. Growth of the number of TV channels and multi-channel platforms in Europe continues despite the crisis, 2009.

[11] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. In *ICDE '99: Proceedings of the 15th International Conference on Data Engineering*, page 512, Washington, DC, USA, 1999. IEEE Computer Society.

[12] I. Guy, N. Zwerdling, I. Ronen, D. Carmel, and E. Uziel. Social media recommendation based on people and tags. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 194–201, New York, NY, USA, 2010. ACM.

[13] G. Harboe, N. Massey, C. Metcalf, D. Wheatley, and G. Romano. The uses of social television. *Computers in Entertainment*, 6(1):1–15, 2008.

[14] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[15] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 195–202, New York, NY, USA, 2009. ACM.

[16] G. N. Lance and W. T. Williams. A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems. *The Computer Journal*, 9(4):373–380, February 1967.

[17] D. Lee and S. Lee. Fast and easy channel-surfing for digital televisions by two buttons and backward regress-able binary tree. *Consumer Electronics, IEEE Transactions on*, 56(3):1880–1882, aug. 2010.

[18] E. Lee, J. Whang, U. Oh, K. Koh, and H. Bahn. Popular channel concentration schemes for efficient channel navigation in Internet Protocol televisions. *Consumer Electronics, IEEE Transactions on*, 55(4):1945–1949, november 2009.

[19] B. Ludwig, S. Mandl, and S. von Mammen. What's on tonight: user-centered and situation-aware proposals for TV programmes. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 258–260, New York, NY, USA, 2006. ACM.

[20] S. B. Michael Fink, Michele Covell. Social- and Interactive-Television Applications Based on Real-Time Ambient-Audio Identification. In *Euro-ITV '06: European Interactive TV Conference*, 2006.

[21] M. Pazzani and D. Billsus. Content-Based Recommendation Systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer Berlin / Heidelberg, 2007.

[22] T. Qiu, Z. Ge, S. Lee, J. Wang, J. Xu, and Q. Zhao. Modeling user activities in a large IPTV system. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 430–441, New York, NY, USA, 2009. ACM.

[23] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, and J. Xu. Modeling channel popularity dynamics in a large IPTV system. In *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 275–286, New York, NY, USA, 2009. ACM.

[24] F. M. Ramos, J. Crowcroft, R. J. Gibbens, P. Rodriguez, and I. H. White. Reducing channel change delay in iptv by predictive pre-joining of tv channels. *Signal Processing: Image Communication*, 26(7):400–12, 2011.

[25] I. Safro, D. Ron, and A. Brandt. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, 60(1):24–41, 2006.

[26] B. Smyth and P. Cotter. Personalized Electronic Program Guides for Digital TV. *AI Magazine*, 22(2):89–98, 2001.

[27] P. H. A. Sneath and R. R. Sokal. *Numerical taxonomy: the principles and practice of numerical classification*. W. H. Freeman and Co, San Francisco, 1973.

[28] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438, 1958.

[29] R. Xu and I. Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, may 2005.

[30] J. Zimmerman, K. Kauapati, A. L. Buczak, D. Schaffer, S. Gutta, and J. Martino. TV Personalization System. In *Personalized Digital Television*, volume 6 of *Human-Computer Interaction Series*, pages 27–51. Springer Netherlands, 2004.