Clustered segmentations

Aristides Gionis Heikki Mannila Evimaria Terzi

HIIT, Basic Research Unit
Department of Computer Science
University of Helsinki, Finland
lastname@cs.helsinki.fi

ABSTRACT

The problem of sequence and time-series segmentation has been discussed widely and it has been applied successfully in a variety of areas, including computational genomics, data analysis for scientific applications, and telecommunications. In many of these areas the sequences involved are multidimensional, and the goal of the segmentation is to discover sequence segments with small variability. One of the characteristics of existing techniques is that they force all dimensions to share the same segment boundaries, yet, it is often reasonable to assume that different dimensions are more correlated than others, and that concrete and meaningful states are associated only with a subset of dimensions.

In this paper we study the problem of segmenting a multidimensional sequence when the dimensions of the sequence are allowed to form clusters and be segmented separately within each cluster. We demonstrate the relevance of this problem to many data-mining applications. We discuss the connection of our setting with existing work, we show the hardness of the suggested problem, and we propose a number of algorithms for its solution. Finally, we give empirical evidence showing that our algorithms work well in practice and produce useful results.

1. INTRODUCTION

Methods for segmenting sequence and time-series data have been discussed widely in the area of data mining. The goal of the segmentation problem is to discover sequence segments with small variability, and segmentation algorithms have been a key to compact representation and knowledge discovery in sequential data. A variety of segmentation algorithms have been proposed and they have been used successfully in many application areas, including computational genomics, data analysis for scientific applications, and telecommunications [8, 10, 14, 17]. In the next paragraphs we discuss three concrete examples in which different kinds of segmentation methods have been applied effectively; these examples also motivate the work presented in this paper.

Example 1. Himberg et al. [10] demonstrated the applicability of sequence-segmentation algorithms for the problem of "context awareness" in the area of mobile communications. The notion of context awareness can be a very powerful cue for improving the friendliness of mobile devices. As a few examples consider the situations where a mobile device adjusts automatically the ring tone, the audio volume, the screen font size, and other controls depending on where the users are located, what they are doing at that time, who else is around, what is the current noise or temperature level, and numerous other such context variables. The approach followed by Himberg et al. [10] is to infer context information from sensors attached to mobile devices: sensors for acceleration, noise level, temperature, luminosity, etc. Measurements from these sensors form naturally multidimensional time series. "Context" can then be inferred by segmenting the time series and annotating the various segments with concrete state descriptions (for example, if ACCELERATION $\approx u_0$, NOISE $\approx v_0$, and ILLUMINATION $\approx w_0$, then STATE = WALKINGINTHESTREET).

Example 2. The problem of discovering *recurrent sources* in sequences is examined in [8]. The idea is that many genomic (or other multivariate) sequences are often assembled by a small number of possible "sources", each of which might contribute several segments in the sequence. For instance, Azad et al. [2] try to identify a coarse-grained description of a given DNA string in terms of a smaller set of distinct domain labels. The work in [8] extends existing sequence-segmentation algorithms in a way that the resulting segments are associated with a description label, and the same label might appear in several segments of the sequence.

Example 3. One of the most important discoveries for the search of structure in genomic sequences is the "block structure" discovery of haplotypes. To explain this notion, consider a collection of DNA sequences over n marker sites (e.g., SNPs) for a population of p individuals. The "haplotype block structure" hypothesis states that the sequence of markers can be segmented in blocks, so that, in each block most of the haplotypes in the population fall into a small number of classes. The description of these haplotypes can be used for further knowledge discovery, e.g., for associating specific blocks with specific genetic-influenced diseases [9].

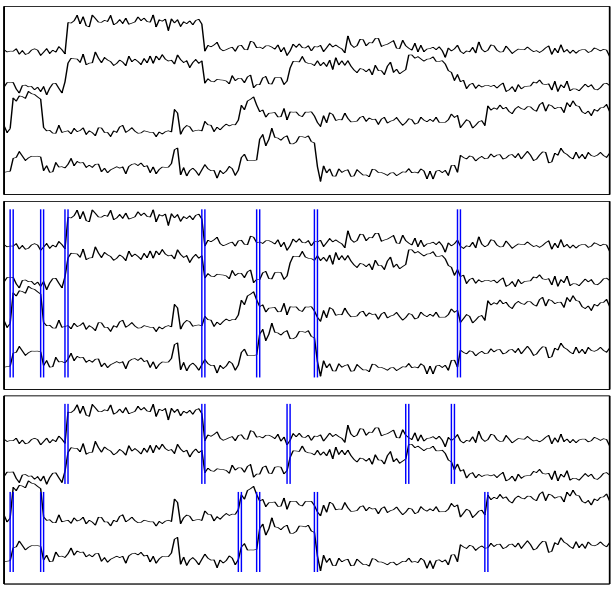


Figure 1: An illustrating example.

From the computational point of view, the problem of discovering haplotype blocks in genetic sequences can be viewed as partitioning a long multidimensional sequence into segments, such that, each segment demonstrates low diversity among the different dimensions. Naturally, segmentation algorithms have been applied to a good effect for this problem [17, 6, 20, 22].

In all of the above examples, segmentation algorithms have been employed for discovering the underlying structure of multi-dimensional sequences. The goal is to find segments with small variability. In previous approaches, however, dimensions are typically forced to share the same segment boundaries. On the other hand, it is often reasonable to assume that different dimensions are more correlated than others, and that concrete and meaningful states are associated with only small subsets of the dimensions.

An illustrating example of the above assumption is shown in Figure 1. The input sequence, a four-dimensional time series, is shown in the top box. In the middle box, it is shown a globally optimal segmentation for the input time series when all dimensions share common segment boundaries. In this example, one can see that the global segmentation provides a good description of the sequence—in all dimensions most of the segments can be described fairly well using a constant value. However, some of the segments are not quite uniform, while on the other hand there are some segment boundaries introduced in relatively constant pieces of the sequence. These representation problems can be alleviated if one allows different segment boundaries among subsets of dimensions, as shown in the lower box of Figure 1. We see that the segmentation after clustering the dimensions in pairs $\{1, 2\}$ and $\{3, 4\}$ gives a "tighter" fit and a more intuitive description of the sequence, even though the number of segments used for each cluster is smaller than the number of segments used in the global segmentation.

In this paper we study the problem of segmenting a multidimensional sequence when subsets of dimensions are allowed to be clustered and segmented separately from other subsets. We call this problem *clustered segmentation*.

Clustered segmentation can be used to extend and improve the quality of results in all of the applications mentioned in our motivating examples: In the application of context awareness, certain context states might be independent of some sensor readings, therefore, a segmentation based on all sensors simultaneously would be a bad predictor. For the problem of discovering recurrent segments one can imagine situations that sources are associated only with a subset of dimensions, for instance, specific regions in DNA sequences (CpG-islands) are associated with a low-dimensional signal (concentration of C and G bases) out of the whole genomic "vocabulary". Finally, in the problem of haplotype-block discovery, the sequence dimensions correspond to different individuals of the population, and thus, clustering of the dimensions would allow the discovery of subpopulation groups with distinct haplotype block structure. The existence of such subpopulations gives rise to a *mosaic structure* of haplotypes, which is a viable biological hypothesis [23, 25].

For solving the problem of clustered segmentation, measures and algorithms that give more refined segmentations and describe the data accurately need to be found. In this paper we are making the following contributions:

- We define the problem of clustered segmentation, and we demonstrate its relevance with existing data-mining applications.
- We demonstrate the hardness of the clustered segmentation problem and we develop practical algorithms for its solution.
- We perform extensive experimental evaluation on synthetically generated and real data sets. Our experiments study the behavior of the suggested algorithms. Furthermore, for many cases of real data sets, we show that clustered segmentation can provide a better model for representing and understanding the data.

The rest of this paper is organized as follows. In the next Section we define the problem of clustered segmentation in detail and we discuss its connection with other known problems. In Section 3 we describe a number of algorithms for solving the problem of clustered segmentation, and in Section 4 we describe our experiments. Finally, Section 5 is a short conclusion.

2. DESCRIPTION OF THE PROBLEM

As we discussed in the introduction, the goal of this paper is to develop improved measures and algorithms for the problem of multi-dimensional sequence segmentation. We now describe the problem in detail by first introducing the necessary notation. Let $S = \{s_1, \ldots, s_d\}$ be a d-dimensional sequence, where s_i is the i-th dimension (signal, attribute, individual, etc.). We assume that each dimension is a sequence of n *values*, and we denote by $s_i[u]$ the value at the u-th *position* of s_i. We write $S' \sqsubseteq S$ to denote that S' is a

subsequence of S, and $s_i[u, v]$ is the subsequence of s_i between the positions u and v. Similarly we denote by $S[u, v]$ the subsequence of S between the positions u and v when all dimensions are considered. The positions on each dimension are naturally *ordered*, for example, in time-series data the order is induced by the time attribute, while in genomic data the order comes from the position of each base in the DNA sequence. In addition, we assume that the dimensions of the sequence are *aligned*, that is, the values at the u-th position of all dimensions are semantically associated (e.g., they correspond to the same time).

We denote by σ a *k-segmentation* of the sequence $\langle 1, \ldots, n\rangle$, that is, a partitioning of $\langle 1, \ldots, n\rangle$ into k contiguous and non overlapping segments. As a result, for specifying a k-segmentation it suffices to provide the $k-1$ boundary points. We write a k-segmentation as $\sigma = \langle \sigma[1], \ldots, \sigma[k]\rangle$, so that we can refer to the individual segments. Because $\sigma[t]$ refers only to a pair of boundary points, we use the notation $S|\sigma[t]$ to "extract" from S the subsequence between the boundary points of $\sigma[t]$. In other words, the boundary points of a segmentation can be viewed as operators that when "applied" to a sequence they provide the actual subsequence segments. The set of all k-segmentations, i.e., all possible subsets of $k-1$ boundary points out of the n positions, is denoted by $\mathcal{S}_k$.

For accessing the quality of a segmentation of a sequence S we need a measure of the *variability* of each segment of S. Let $f(S')$ be such a measure defined for all $S' \sqsubseteq S$. We typically assume that f is an easily computable cost function. For example, for real-valued sequences, f is often taken to be the variance of the values in S'. Now, the quality of a k-segmentation $\sigma = \langle \sigma[1], \ldots, \sigma[k]\rangle$ on a sequence S is defined to be

$$f(S|\sigma) = \sum_{t=1}^{k} f(S|\sigma[t]),$$

that is, the sum of the cost function f over all segments of S defined by σ. The *sequence-segmentation* problem is to compute the optimal such k-segmentation

$$\sigma^* = \arg\min_{\sigma \in \mathcal{S}_k} f(S|\sigma).$$

For a wide variety of cost functions, a dynamic programming (e.g., algorithm [3]) can be used for computing the optimal k-segmentation.

In the above formulation, the number of segments k is given in advance. If k is variable, the trivial n-segmentation can typically achieve a zero cost. Thus, a popular way for allowing k to be variable is to add a penalization factor for choosing large values of k, for example, the optimal segmentation is now defined to be

$$\sigma^* = \arg\min_{k, \sigma \in \mathcal{S}_k} f(S|\sigma) + k\gamma.$$

A Bayesian approach for selecting the penalization term to make it proportional to the *description length* [21] of the *segmentation model*. For instance, by assuming that for each segment we need to specify one boundary point and d values (one per dimension), one can choose $\gamma = (d+1)\log(dn)$. An important observation, however, is that the same dynamic programming algorithm with no additional time overhead can be used to compute the optimal segmentation for the variable-k version of the problem. In the following and mainly for clarity of exposition, we will only concentrate on the fixed-k version, however, all of our definitions and algorithms can be applied to the variable-k version, as well.

We now proceed to define the *clustered* version of segmentation problems, which is the focus of our paper. As we explained in the introduction, the main idea is to allow subsets of dimensions to be segmented separately. For the next definition we use the notation $s_i|\sigma$ as we used $S|\sigma$: the segmentation σ is applied to the one-dimensional sequence s_i as it was applied to the multi-dimensional sequence S; the cost function $f(s_i|\sigma)$ is computed accordingly.

PROBLEM 1. *Given a d-dimensional sequence S with n values in each dimension, a cost function f defined on all subsequences and all dimensions of S, and integers k and c, compute c different k-segmentations $\sigma_1, \ldots, \sigma_c$, so as to minimize the sum*

$$\sum_{i=1}^{d} \min_{1 \le j \le c} f(s_i|\sigma_j).$$

In other words, we seek to partition the sequence dimensions in c clusters, and compute the optimal segmentation in each one of those, in a way that the total error is minimized. As before, one can use the Bayesian approach to define the variable-c version of the problem, where the optimal value for c is sought, but again we assume that the value of c is given.

2.1 Connections to related work

The clustered segmentation problem, as stated above, is clearly related with the time-series clustering problem [24, 13]. Several definitions for time-series similarity have been discussed in the literature, for example, see [4, 7, 1]. A key difference, however, is that in our formulation signals are assigned to the same cluster if they can be segmented "well" together, while most time-series clustering algorithms base their grouping criteria in a more geometric notion of similarity. To emphasize the difference, we note that our methods can be used to segment non-numerical sequences like the SNP data we described in the introduction.

The formulation in Problem 1 suggests that one can consider clustered segmentation as a k-median type of problem (e.g., see [18]). However, a main difficulty with trying to apply k-median algorithms in our setting, is that the space of solutions is extremely large. Furthermore, for k-median algorithms it is often the case that a "discretization" of the solution space can be applied (seek for solutions only among the input points). Assuming the triangle inequality, this discretization degrades the quality of the solution by a factor of at most 2. In our setting, however, the solution space (segmentations) is different from the input space (sequence), and also many natural distance functions between sequences and segmentations does not form a metric.

Finally, our problem is also related with the notion of *segmentation problems* as introduced by Kleinberg et al. [16].

In [16], starting from an optimization problem, the "segmented" version of that problem is defined by allowing the input to be partitioned in clusters, and considering the best solution for each cluster separately. To be precise, in the terminology of [16] our problem should be called "segmented segmentation" since in our case the optimization problem is the traditional segmentation problem. Even when starting from very simple optimization problems, their corresponding segmented versions turn out to be hard.

2.2 Problem complexity

In this subsection we demonstrate the hardness of the clustered segmentation problem. In our case, the optimization problem we start with is the traditional non-clustered segmentation problem. This problem is solvable in polynomial time when the k-segmentation variance is considered. Not surprisingly the corresponding clustered segmentation is an NP-hard problem.

THEOREM 1. *Clustered segmentation, as defined in Problem 1, with real-valued sequences, and cost function f the variance function, is NP-hard.*

The proof of the theorem can be found in the Appendix.

3. ALGORITHMS

In this section we describe two classes of algorithms for solving the clustered segmentation problem. In the first class we define distance measures between sequence segmentations and we employ a standard clustering algorithm (e.g., k-means) on the pair-wise distance matrix. The second class consists of two randomized algorithms that cluster sequences using segmentations as "centroids". In particular, we use the notion of a distance between a segmentation and a sequence, which is the error induced to the sequence when the segmentation is applied to it. The algorithms of the second class treat the clustered-segmentation problem as a model selection problem and they try to find the best model that describes the data.

Before proceeding with the description of the algorithms, we briefly review the dynamic-programming algorithm that segments a sequence S into k segments. This algorithm is used by almost all of our methods. The idea of the dynamic programming algorithm is to compute the segmentation incrementally, for all subsequences $S[1,i]$ with $i = 1,\ldots,n$, and all values for an l-segmentation with $l = 1,\ldots,k$. In particular, the computation of an l-segmentation σ for the subsequence $S[1,i]$ is based on the equation

$$f(S[1,i]|\sigma) = \min_{\tau \in \mathcal{S}_{l-1},\, 1 \le j < i} f(S[1,j]|\tau) + f(S[j+1,i]).$$

The running time of the dynamic programming algorithm is $O(n^2 dk F(n))$, where $F(t)$ is the time required to compute the function f on a subsequence of length t. In the case that f is the variance function, the computation can be done in constant time (by precomputing the sum of values and the sum of squares of values of all prefixes of the sequence), so the running time of the dynamic programming algorithm is $O(n^2 dk)$.

3.1 Distance-based clustering of segmentations

In this section we define distance functions between the segmentations of two sequences. Such distance functions can be used to construct a pair-wise distance matrix between the dimensions of the sequence. The distance matrix is then used for clustering the dimensions via a standard distance-based clustering algorithm; in our case, we use standard the k-means algorithm. k-means despite its limitations as a hill-climbing algorithm that is not guaranteed to converge to a global optimum, is mainly used because it is efficient and it works very well in practice. Variations of the k-means algorithm have been proposed for time-series clustering, for example in [24].

The main idea of the k-means algorithm is the following: Given N points that need to be clustered and a distance function d between them, the algorithm starts by selecting k random points as cluster centers and assigning the rest of the $N-k$ points to the closest cluster center, according to d. In that way k clusters are formed. Within each cluster the mean of the points defining the cluster is evaluated and the process continues iteratively with those means as the new cluster centers, until convergence.

The two distance functions defined here are rather intuitive and simple. The first one, D_E, is based on the mutual exchange of optimal segmentations of the two sequences and the evaluation of the additional error such an exchange introduces. Therefore, two sequences are then similar if the best segmentation of the one describes "well" the second, and vice versa. The second distance function, D_P, is probabilistic and it defines the distance between two sequences by comparing the probabilities of each position in the sequence being a segmentation boundary.

3.1.1 Distance as a measure of fit of optimal segmentations

The goal of our clustering is to cluster together dimensions in such a way that similarly segmented dimensions are put in the same cluster, while the overall cost of the clustered segmentation to be minimized. Intuitively this means that a distance function should perform well if it quantifies how well the optimal segmentation of the one sequence describes the other one and vice versa. Based on exactly this notion of "exchange" of optimal segmentations of sequences, we define the distance function D_E in the following way.

Given two dimensions s_i, s_j and their corresponding optimal k-segmentations $\sigma_i^*, \sigma_j^* \in \mathcal{S}_k$ we define the distance of s_i from σ_j^* denoted by $D_E(s_i, \sigma_i^*|\sigma_j^*)$ as follows:

$$D_E(s_i, \sigma_i^*|\sigma_j^*) = f(s_i|\sigma_j^*) - f(s_i|\sigma_i^*)$$

However in order for the distance between two sequences and their corresponding segmentations to be symmetric we alternatively use the following symmetric definition of D_E:

$$D_E(s_i, \sigma_i^*, s_j, \sigma_j^*) = D_E(s_i, \sigma_i^*|\sigma_j^*) + D_E(s_j, \sigma_j^*|\sigma_i^*)$$

3.1.2 Probabilistic distance

Distance function D_P is based on comparing two dimensions by comparing the probability distributions of their

points being segment boundaries.[1] For each dimension s_i with $1 \leq i \leq d$ we associate a probability distribution p_i. The value of $p_i[t]$ at point t with $1 \leq t \leq n$ corresponds to the probability of the t-th point of the series being a segment boundary. Associating a probability distribution with each sequence, allows us to define the distance between two sequences as the variational distance between the corresponding distributions. Therefore, we define the distance function $D_P(s_i, s_j)$ between the dimensions s_i and s_j as follows:

$$D_P(s_i, s_j) = \mathrm{Var}(p_i, p_j) = \sum_{1 \leq t \leq n} |p_i[t] - p_j[t]| \qquad (1)$$

Computing the probabilities of segment boundaries for a given sequence and given the required k number of segments, can be done in $O(n^2k)$ time using dynamic programming. Consider a dimension s of our d-dimensional sequence. Let the probability that there is a segment boundary at point t of s, when segmented using k-segments. Denote by $\mathcal{S}_k^{(t)}$ the set of all k-segmentations having a boundary at point t. Then we are interested in the probability of any segmentation from the set $\mathcal{S}_k^{(t)}$ given the sequence s:

$$p(\mathcal{S}_k^{(t)}|s) = \sum_{\sigma \in \mathcal{S}_k^{(t)}} p(\sigma|s).$$

Since $\mathcal{S}_k$ refers to the set of all k-segmentations of the full sequence s, the above equation can be rewritten as follows:

$$p(\mathcal{S}_k^{(t)}|s) = \frac{\sum_{\sigma' \in \mathcal{S}_k^{(t)}} p(\sigma', s)}{\sum_{\sigma \in \mathcal{S}_k} p(\sigma, s)}$$

For the joint probabilities of segmentation and sequence, $p(\sigma, s)$, it holds that:

$$p(\sigma, s) = \frac{1}{Z} 2^{-\sum_{[a,b] \in \sigma} f(s[a,b]|\sigma)}.$$

In the above equation Z is a normalizing constant that cancels out. For building the dynamic programming equations we need to define the following entity for a segmentation σ:

$$q(a, b) = 2^{-f(s[a,b]|\sigma)}.$$

Additionally, for any interval $[t, t']$ and considering segmentations consisting of i-segments (where $1 \leq i \leq k$) we define:

$$Q_i(t, t') = \sum_{\sigma \in \mathcal{S}_i[t,t']} \prod_{[a,b] \in \sigma} q(a, b).$$

Since $\mathcal{S}_k[t, t+1]$ is equal to the Cartesian product $\mathcal{S}_i[1, t] \times \mathcal{S}_{k-i}[t+1, n]$ for $1 \leq i \leq k$ we have that:

$$p(\mathcal{S}_k^{(t)}|s) = \sum_{1 \leq i \leq k} \frac{Q_i(1, t) Q_{k-i}(t+1, n)}{Q_k(1, n)}.$$

Finally the inner-most equations for the dynamic programming, that consider segmentations of a fixed number of segments i (with $1 \leq i \leq k$) are:

$$Q_i(1, b) = \sum_{1 \leq a \leq b} Q_{i-1}(1, a-1) q(a, b)$$

and

$$Q_i(a, n) = \sum_{a \leq b \leq n} q(a, b) Q_{i-1}(b+1, n).$$

[1]For a similar development see [17].

Using the above equations we can compute the probabilities of each point being a segment boundary in each one of the d dimensions of S. Pairwise distances between two dimensions are evaluated using Equation (1).

3.2 Non-distance based clustering of segmentations

In this section we describe two algorithms that treat clustered segmentation as a model-selection problem. The first algorithm, SAMPLSEGM, is a sampling algorithm and it is motivated by the theoretical work of Kleinberg et al. [16], Indyk [11, 12] and Charikar et al. [5]. The second, ITERCLUSTSEGM, is an adaptation of the popular k-means algorithm. Both algorithms are simple and intuitive and they perform well in practice.

3.2.1 The SAMPLSEGM algorithm

The basic idea behind the SAMPLSEGM approach is the intuition that if the data exhibit clustered structure, then a small sample of the data would exhibit the same structure. The reason is that for large clusters in the data set one would expect to sample enough data, so that similar clusters appear in the sampled data. On the other hand, one can possibly afford to miss data from small clusters in the sampling process, because small clusters do not contribute much in the overall error function. Our algorithm is motivated by the work of Kleinberg et al. [16], in which a sampling algorithm for the segmented version of the catalog problem is proposed. Similar ideas have been used successfully by Indyk [11, 12] for the problem of clustering in metric spaces.

For the clustered segmentation problem we adopt a natural sampling-based technique: We first sample uniformly at random a small set A of $r \log d$ dimensions, where r is a small constant. Then we search exhaustively all possible *partitions* of A into c clusters $A_1, \ldots, A_c$. For each cluster A_j we find the optimal segmentation $\sigma_j \in \mathcal{S}_k$ for the sequence S on the dimensions that are associated with A_j. The rest of the dimensions s_i that are not included in the sample, are assigned to the set j that minimizes the error $f(s_i|\sigma_j)$. The partition of the sample set A that causes the least error is considered to be the solution found for the set A. The whole sampling process is repeated with different sample sets A for a small number of times (in our experiments 3 times) and the best result is reported as the output of the sampling algorithm.

When the size of the sample set is logarithmic in the number of dimensions, the overall running time of the algorithm is polynomial. In our experiments, we found that the method is accurate for data sets of moderate size, but it does not scale well for larger data sets.

3.2.2 The ITERCLUSTSEGM algorithm

The ITERCLUSTSEGM algorithm is an adaptation of the widely-used k-means algorithm where the cluster means are replaced by the common segmentation of the dimensions in the cluster and the distances of a sequence to the cluster is the error induced when the cluster's segmentation is applied to the sequence.

Therefore in our case, the c centers correspond to c different

segmentations. The algorithm is iterative and at the t-th iteration step it keeps an estimate for the solution segmentations $\sigma_1^t, \ldots, \sigma_c^t$, which is to be refined in the consecutive steps. The algorithm starts with a random clustering of the dimensions, and it computes the optimal k-segmentation for each cluster. At the $(t+1)$-th iteration step, each dimension s_i is assigned to the segmentation σ_j^t for which the error $f(s_i|\sigma_j^t)$ is minimized. Based on the newly obtained clusters of dimensions, new segmentations $\sigma_1^{t+1}, \ldots, \sigma_c^{t+1}$ are computed, and the process continues until there is no more improvement in the error. The complexity of the algorithm is $O(I(cd+cP(n,d)))$, where I is the number of iterations until convergence, and $P(n,d)$ is the complexity of segmenting a sequence of length n and d dimensions.

4. EXPERIMENTS

In this section we describe the experiments we performed in order to evaluate the validity of the clustered segmentation model and the behavior of the suggested algorithms. For our experiments we used both synthetically generated data, as well as real data consisting of time series and genomic sequences. For the synthetic data we report that in all cases the true underlying model, used to generate the data, is found. For the real data we found that in all cases clustered segmentations, output by the proposed algorithms, produce better models than the models produced by non-clustered segmentations.

4.1 Ensuring fairness in model comparisons

In the experimental results shown in this section we report the accuracy in terms of errors. Our intention is to consider the error as a measure of comparing models: a smaller error indicates a better model. However, this is the case when the compared models have the same number of parameters. It would be unfair to compare the errors induced by two models with different number of parameters, because in this case the trivial model of each point described by itself would induce the the least error and would be the best.

Therefore, to make the comparison between two different models fair, we are taking care to ensure that the same number of parameters is used in both models. We briefly describe the methodology we followed in order to guarantee fairness in comparing the different models. Consider a k-segmentation for a d-dimensional sequence S of length n. If no clustering of dimensions is considered, the number of parameters that are necessary to describe this k-segmentation model is $k(d+1)$. This number comes from the fact that we can describe the model by specifying the starting point and the d mean values—one for each dimension—for each one of the k segments. Consider now a clustered segmentation of the sequence with c clusters and k' segments for each cluster. The number of parameters for this model is $d + \sum_{i=1}^{c} k'(d_i+1) = d + k'(d+c)$, since, in addition to specifying the starting points and the values for each cluster, we also need d parameters to indicate the cluster that each dimension belongs to. In our experiments, in order to compare the errors induced by the two models we select parameters so that $k(d+1) = d + k'(d+c)$.

4.2 Experiments on synthetic data

We first describe our experiments on synthetic data. For the purpose of this experiment, we have generated sequence data from a known model, and the task is to test if the suggested algorithms are able to discover that model.

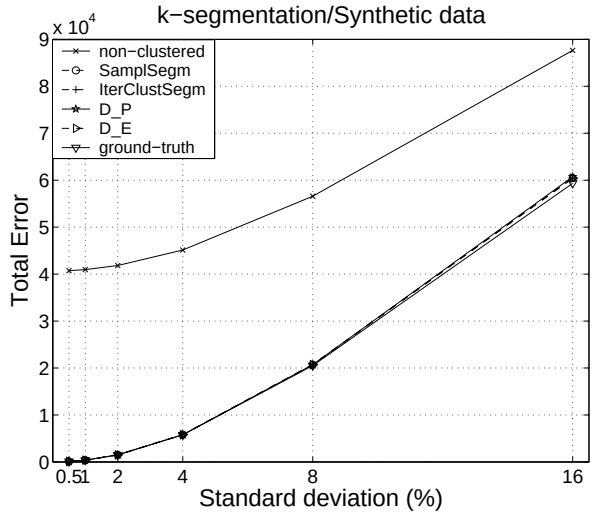
The data we used were generated as follows: the d dimensions of the generated sequence were divided in advance into c clusters. For each cluster we select k segment boundaries, which are common for all the dimensions in that cluster, and for the j-th segment of the i-th dimension we select a mean value μ_{ij}, which is uniformly distributed in $[0,1]$. Points are then generated by adding a noise value sampled from the normal distribution $\mathcal{N}(\mu_{ij}, \sigma^2)$. An example of a small data set generated by this method is shown in Figure 1. For our experiments we fixed the values $n = 1000$ points, $k = 10$ segments, and $d = 200$ dimensions. We created different data sets using $c = 2, ..., 6$ clusters and with standard deviations varying from 0.005 to 0.16.

The results for the synthetically generated data are shown in Figure 2. One can see that the errors of the reported clustered segmentation models are typically very low for all of our algorithms. In most of the cases all proposed methods approach the *true* error value. Here we report the results for small sample sizes (usually $(c+4)$ samples with c being the number of clusters). Since our algorithms are randomized we repeat each one of them for 5 times and report the best found solution. Apart from the errors induced by the proposed algorithms, the figures include also two additional errors. The error induced by the non-clustered segmentation model with the same number of parameters and the error induced by the true model that has been used for generating the data ("ground-truth"). The first one is always much larger than the error induced by the models reported by our algorithms. In all the comparisons between the different segmentation models we take into consideration the fairness criterion discussed in the previous subsection.

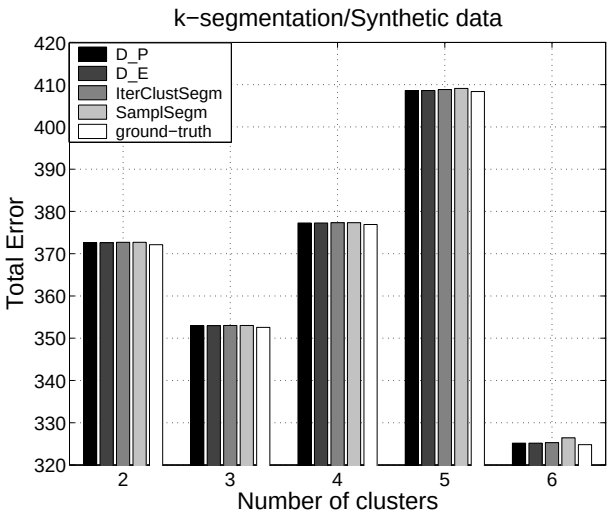
As indicated in Figure 2(a) the difference in errors becomes smaller as the standard deviation increases. This is natural since as standard deviation increases all dimensions tend to become uniform and the segment structure disappears. The error of the clustered segmentation model for different number of clusters is shown in Figure 2(b). The better performance of the clustered model is apparent. Notice that the error caused by the non-clustered segmentation is an order of magnitude larger than the corresponding clustered segmentation results and thus omitted from the plot.

4.3 Experiments on time-series data

Next, we tested the behavior of the clustered segmentation model on real time-series data sets obtained by the UCR time-series data mining archive [15]. We used the **phone** and the **spot_exrates** data sets of the archive. The **phone** data set consists of 8 dimensions each one corresponding to the value of a sensor attached to a mobile phone. For the clustered segmentation we used number of segments $k = 8$ and number of clusters $c = 2$, 3 and 4, while for the non-clustered segmentation we used $k = 10$, 11, and 11, respectively so that we again guarantee fair comparison. Figure 3(a) shows the error induced by the clustered and the non-clustered segmentations for different number of clusters.

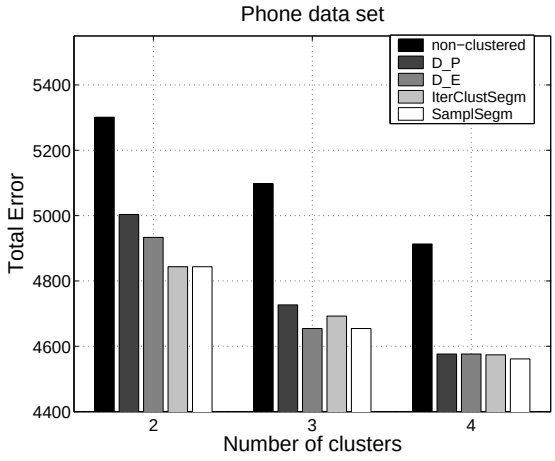


(a) Total error of the model as a function of standard deviation in the generated data.

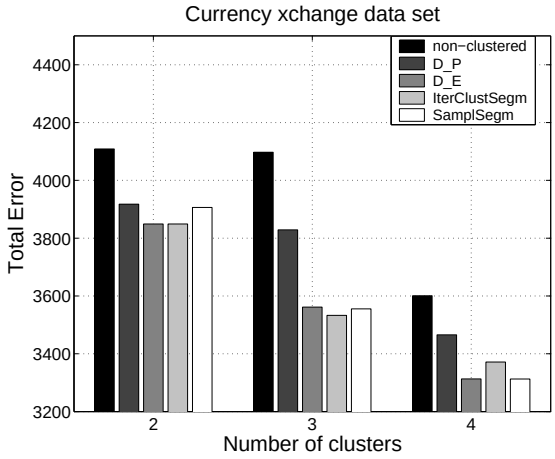


(b) Total error as a function of the number of clusters existing in the generated data. The bar that corresponds to the non-clustered model is missing since it is an order of magnitude larger.

Figure 2: Error for k-segmentations on synthetic data sets.



(a)



(b)

Figure 3: Experiments with real time-series data.

Apparently, the clustered segmentation model reduces by far the induced error. SamplSegm, IterClustSegm and clustering using D_E are all giving the same level of error, with clustering using D_P performing almost equally well, and in all cases better than the non-clustered segmentation.

Analogous results are obtained for the **spot_exrates** data set as illustrated in Figure 3(b). This data set contains the spot prices (foreign currency in dollars) and the returns for daily exchange rates of the 12 different currencies relative to the US dollar. There are 2567 (work-)daily spot prices, and so 2566 daily returns for each of these 12 currencies, over a period of about 10 years (10/9/86 to 8/9/96). Again, we considered the case of k-segmentations. For the clustered segmentation we used number of segments $k = 10$ and number of clusters $c = 2$, 3, and 4, while for the non-clustered segmentation we used $k = 12$, 12 and 13, respectively.

4.4 Experiments on mobile-sensor data

Finally, we tested the behavior of the proposed methods on the benchmark dataset for context recognition described in [19].[2] The data were recorded using microphones and a sensor box, attached to a mobile phone. The combination was carried by the users during the experiments and the data were logged by a laptop carried by the users. The signals collected were transformed into 29 variables the values of which have been recorded for some periods of time.

The dataset basically contains 5 scenarios each one repeated for a certain number of times. The 29 variables recorded, that correspond to the 29 dimensions, are related to *device position*, *device stability*, *device placement*, *light*, *temperature*, *humidity*, *sound level* and *user movement*.

The results of the clustered segmentations algorithms for scenario 1 and 2 are shown in Figures 4 and 5. For the rest of the scenarios the results are similar and thus omitted. Since some dimensions in the different scenarios are all constant we have decided to ignore them. Therefore from a total of 29 dimensions we have considered 20 for scenario 1, 19 for scenario 2, 16 for scenario 3, 14 for scenario 4 and 15 for scenario 5.

For the case of clustered segmentation we considered $k = 5$ and $c = 2, 3, 4, 5$ for all the scenarios, while the corresponding values of k for the non-clustered segmentation that could guarantee fairness of comparison of the results was evaluated to be $k = 6, 7$ depending on the value of c and the number of dimensions in the scenario. The error levels using the different methods proposed here, as well as the non-clustered segmentation algorithm are shown in Figures 4 and 5. In all cases the clustered segmentation model found by any of our four methods has much lower error level than the corresponding non-clustered one. Some indicative clusterings of the dimensions of scenario 1 using the proposed methods are shown in Figure 6. Notice that the clustering shown in Figure 6 reflects an expected clustering of dimensions. The first cluster contains only dimensions related to the "Position", the "Stability" and the "Placement" of the device. The second cluster puts together all time series related to the "Humidity" of the environment. The third cluster consists of dimensions related to the "Light" and the "Sound Pressure" of the environment as well as the user movement. Finally the last cluster contains all the dimensions related to the "Temperature" of the environment as well as the dimensions that corresponds to the "Running" dimensions that characterizes the user movement. Although this last results raises some suspicious, since running is the only user action related dimension that is clustered separately from the other two, it can be quite easily explained if we observe Figure 8. It is obvious that segmentation-wise dimensions "UserAction:Movement:Walking" and "'UserAction:Movement:WalkingFast" are much closer to each other than they are with "UserAction:Movement:Running", while the latter one can be easily segmented using segmentation boundaries of dimensions "Environment:Temperature:Warm" and "Environment:Temperature:Cool". Similar observations can be made also for the rest of the clusterings obtained using the other three proposed methods. Indicatively we show

[2]The dataset is available at
http://www.cis.hut.fi/jhimberg/contextdata/index.shtml

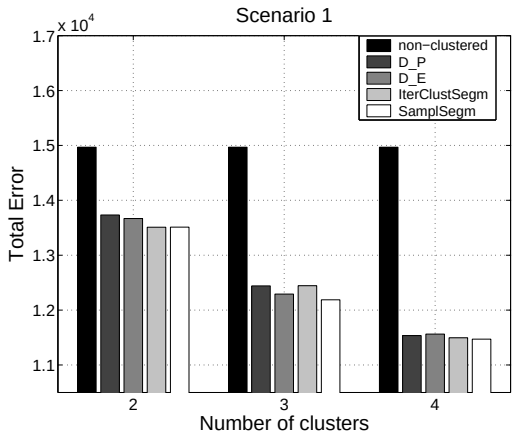


Figure 4: Experiments with scenario 1.

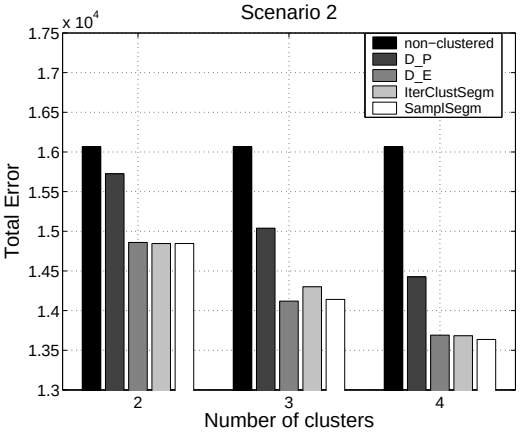


Figure 5: Experiments with scenario 2.

in Figure 7 the clustering obtained using k-means algorithm for the same number of clusters and using L_1 as the distance metric between the different dimensions. There is not an obvious correspondence between the clustering found in that case and the expected clustering with respect to the general categories each dimension belongs to.

4.5 Discussion

The experimental evaluation performed on both synthetic and real data indicates that the clustered segmentation model is a more precise alternative for describing the data at hand, and all the proposed methods find models that show much smaller error than the error of the equivalent non-clustered segmentation model, in all the considered cases. Overall, in the case of synthetic data the true underlying model, used for the data generation, is always found. For the real data, the true model is unknown and thus we base our conclusions on the errors induced by the two alternative models when the same number of parameters is used.

The proposed algorithms are intuitive and they perform well in practice. For most of the cases they give equivalent results and they find almost the same models. Noticeably, SamplSegm appears to be competitive in terms of quality of

Device:Position:DisplayDown, Device:Position:AntennaUp, Device:Stability:Stable, Device:Stability:Unstable, Device:Placement:AtHand
Environment:Humidity:Humid, Environment:Humidity:Normal, Environment:Humidity:Dry
Environment:Light:EU, Environment:Light:Bright, Environment:Light:Normal, Environment:Light:Dark, Environment:Light:Natural, Environment:SoundPressure:Silent, Environment:SoundPressure:Modest, UserAction:Movement:Walking, UserAction:Movement:WalkingFast
Environment:Temperature:Warm, Environment:Temperature:Cool, UserAction:Movement:Running

Figure 6: Clustering of the dimensions of Scenario 1 into 4 clusters using ITERCLUSTSEGM algorithm.

Environment:Humidity:Dry
Environment:Light:Bright, Environment:Light:Natural, UserAction:Movement:WalkingFast
Device:Position:AntennaUp, Device:Stability:Unstable, Environment:Light:EU, Environment:Light:Normal, Environment:Temperature:Warm, Environment:Humidity:Humid, Environment:SoundPressure:Silent, 'UserAction:Movement:Walking
Device:Position:DisplayDown, Device:Stability:Stable, Device:Placement:AtHand, Environment:Light:Dark, Environment:Temperature:Cool, Environment:Humidity:Normal, Environment:SoundPressure:Modest, UserAction:Movement:Running'

Figure 7: Clustering of the dimensions of Scenario 1 into 4 clusters using L_1 distance k-means.

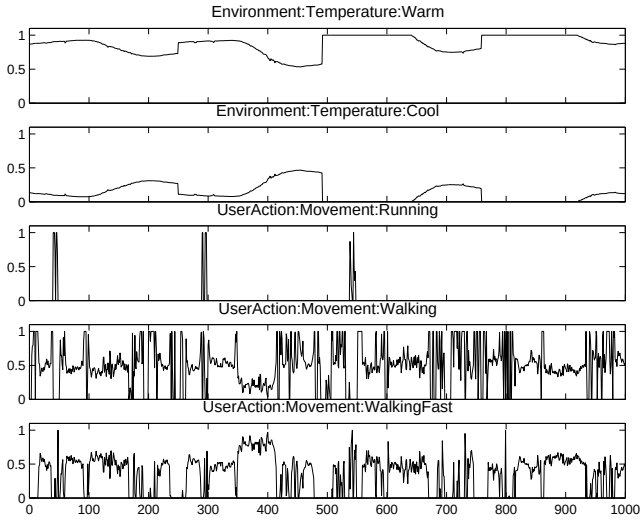


Figure 8: Subset of dimensions of Scenario 1 dataset that explain clustering in Figure 6.

results even for small sample sizes. However, it can become slow as the sample size increases since we need to examine all possible partitions of the sample and select the one that implies the best model. On the other hand, the other three alternatives scale smoothly and appears to perform well in practical settings.

5. CONCLUSIONS

We have introduced the clustered segmentation problem where the task is to cluster the dimensions of a multidimensional sequence into c clusters so that the dimensions grouped in the same cluster share the same segmentation points. The problem when considered for real-valued sequences and cost function the variance function is NP-hard. We described simple algorithms for solving the problem. All proposed methods perform well for both synthetic and real data sets consisting of time-series data. In all the cases we experimented with, the clustered segmentation model seems to describe the datasets better than the corresponding non-clustered segmentation model with the same number of parameters.

6. REFERENCES

[1] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceedings of the 21st International Conference on Very Large Data Bases*, pages 490–501, Zurich, Switzerland, 1995.

[2] R. K. Azad, J. S. Rao, W. Li, and R. Ramaswamy. Simplifying the mosaic description of DNA sequences. *Physical Review E, 66, article 031913*, 2002.

[3] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6), 1961.

[4] B. Bollobas, G. Das, D. Gunopulos, and H. Mannila. Time-series similarity problems and well-separated

geometric sets. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 454–456, 1997.

[5] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 30–39, 2003.

[6] M. J. Daly, J. D. Rioux, S. F. Schaffner, T. J. Hudson, and E. S. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29:229–232, 2001.

[7] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time series databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 419–429, 1994.

[8] A. Gionis and H. Mannila. Finding recurrent sources in sequences. In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology*, pages 115–122, Berlin, Germany, 2003.

[9] D. Gusfield. Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *International Conference on Research in Computational Molecular Biology*, pages 166–175, 2002.

[10] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. Toivonen. Time series segmentation for context recognition in mobile devices. In *Proceedings of the IEEE International Conference on Data Mining*, pages 203–210, 2001.

[11] P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 428–434, 1999.

[12] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 154–159, 1999.

[13] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of arima time-series. In *Proceedings of the IEEE International Conference on Data Mining*, pages 273–280, 2001.

[14] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the IEEE International Conference on Data Mining*, pages 289–296, 2001.

[15] E. Keogh and T. Folias. The UCR time series data mining archive, 2002.

[16] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Segmentation problems. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 473–482, 1998.

[17] M. Koivisto, M. Perola, T. Varilo, et al. An MDL method for finding haplotype blocks and for estimating the strength of haplotype block boundaries. In *Pacific Symposium on Biocomputing*, pages 502–513, 2003.

[18] J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[19] J. Mantyjarvi, J. Himberg, P. Kangas, U. Tuomela, and P. Huuskonen. Sensor signal data set for exploring context recognition of mobile devices. In *In Workshop Benchmarks and a database for context recognition in conjuction with the 2nd Int. Conf. on Pervasive Computing (PERVASIVE 2004)*, 2004.

[20] N. Patil, A. J. Berno, D. A. Hinds, et al. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294:1669–70, 2001.

[21] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[22] M. Salmenkivi, J. Kere, and H. Mannila. Genome segmentation using piecewise constant intensity models and reversible jump MCMC. In *Proceedings of the European Conference on Computational Biology*, pages 211–218, 2002.

[23] R. Schwartz, B. V. Halldorsson, V. Bafna, A. G. Clark, and S. Istrail. Robustness of inference of haplotype block structure. *Journal of Computational Biology*, 10(1):13–9, 2003.

[24] M. Vlachos, J. Lin, E. Keogh, and D. Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series, 2003.

[25] J. D. Wall and J. K. Pritchard. Assessing the performance of the haplotype block model of linkage disequilibrium. *American Journal of Human Genetics*, 73:502–515, 2003.

Appendix

THEOREM 1. *Clustered segmentation, as defined in Problem 1, with real-valued sequences, and cost function f the variance function, is NP-hard.*

PROOF. We give only an outline of the proof idea. The problem from which we obtain the reduction is the SET COVER, a well-known NP-hard problem. An instance of the SET COVER specifies a ground set U of n elements, a collection $\mathcal{C}$ of m subsets of U, and a number c. The question is whether there are c sets in the collection $\mathcal{C}$ whose union is the ground set U.

Given an instance of the SET COVER, we create an instance of the clustered k-segmentation as follows: We form a sequence with n dimensions. In each dimension there are $2m + 1$ "runs" of values and each run has an equal number of values. There are two types of runs: `High` and `Low`, and these two types are alternating across the sequence. All `High` runs have all their values equal to 1. The `Low` runs are indexed from 1 to m and they in turn can be of two types: Z and E. Z-`Low` runs have all their values equal to 0. E-`Low` runs are split in two pieces, so that the f function on such a run incurs

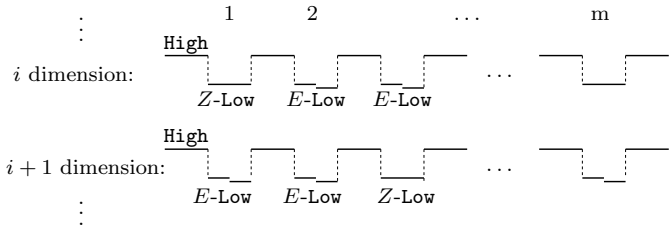


Figure 9: The construction used in the proof of Theorem 1.

cost exactly ϵ. The construction can be seen schematically in Figure 9.

The instance of the SET COVER is encoded in the `Low` runs. If the j-th set of $\mathcal{C}$ contains the i-th element of U, then the j-th `Low` run of the i-th dimension is set to type E, otherwise it is set to type Z. Assume that in total there are L `Low` runs of type E. We would ask for a k-segmentation with $k = 2m + 2$ segments. By setting ϵ to be very small, the $2m + 1$ segments would be separating the `High` from the `Low` runs, and there would be the freedom to segment one more E-`Low`-type run in order to save an additional cost of ϵ. The question to ask is if there is a clustered segmentation with c clusters that has cost at most $(L - n)\epsilon$. One can show that is possible if and only if there is a solution to the original SET COVER problem. Furthermore, ϵ needs only to be polynomial in $1/n$ and $1/m$, so the transformation can be computed in polynomial time. □