Discriminating Subsequence Discovery for Sequence Clustering*

Jianyong Wang[†], Yuzhou Zhang[‡], Lizhu Zhou[§]

Tsinghua University, Beijing, 100084, China

{[†]jianyong, [§]dcszlz}@tsinghua.edu.cn, [‡]zhangyz04@mails.tsinghua.edu.cn

George Karypis

University of Minnesota, Minneapolis, MN 55455, USA

karypis@cs.umn.edu

Charu C. Aggarwal

IBM T.J. Watson Research Center, Hawthorne, NY 10532, USA

charu@us.ibm.com

Abstract

In this paper, we explore the discriminating subsequence-based clustering problem. First, several effective optimization techniques are proposed to accelerate the sequence mining process and a new algorithm, CONTOUR, is developed to efficiently and directly mine a subset of discriminating frequent subsequences which can be used to cluster the input sequences. Second, an accurate hierarchical clustering algorithm, SSC, is constructed based on the result of CONTOUR. The performance study evaluates the efficiency and scalability of CONTOUR, and the clustering quality of SSC.
Keywords. Sequence mining, summarization subsequence, clustering.

1 Introduction

A **sequence** S_i is an ordered list of events, denoted by $\langle e_{i1}, e_{i2}, \ldots, e_{im} \rangle$ (or $e_{i1}e_{i2}\ldots e_{im}$ for brevity), where e_{ij} ($\forall j, 1 \leq j \leq m$) is an event belonging to a set of distinct events, $E = \{e_1, e_2, \ldots, e_n\}$. In this study, any event in E is atomic (thus, we can alternatively call it an item), and can be repetitive in a sequence. The length of a sequence S_i is defined as the number of events in S_i, and a sequence with length l is called an l-sequence. An l-sequence $S_\alpha = \alpha_1 \alpha_2 \ldots \alpha_l$ is said to be *contained* in a k-sequence $S_\beta = \beta_1 \beta_2 \ldots \beta_k$, if $l \leq k$ and there exist integers $1 \leq o_1 < o_2 < \ldots < o_l \leq k$ such that $\alpha_1 = \beta_{o_1}, \alpha_2 = \beta_{o_2}, \ldots, \alpha_l = \beta_{o_l}$. If sequence S_α is contained in sequence S_β, S_α is called a **subsequence** of S_β and S_β a **supersequence** of S_α. This is denoted by $S_\alpha \sqsubseteq S_\beta$ or equivalently $S_\beta \sqsupseteq S_\alpha$.

An input sequence database SDB contains a set of input sequences, and the number of sequences in SDB is called the **base size** of SDB, denoted by $|SDB|$. Given a sequence S_α, the number of input sequences in SDB that contain S_α is called the **absolute support** of sequence S_α w.r.t. sequence database SDB, denoted by $sup^{SDB}(S_\alpha)$. The percentage of input sequences in SDB that *contain S_α* (i.e., $sup^{SDB}(S_\alpha)/|SDB|$) is called the **relative support** of S_α. Sequence S_α is said to be **frequent** if $sup^{SDB}(S_\alpha) \geq min\_sup$, where min_sup is a user-specified absolute support threshold.

As a sequence can be used to naturally model the temporal ordering relationship among a set of events, and abundant sequence data have emerged in recent years such as DNA string, protein sequence, Web log data, and so on, pattern discovery from sequence databases has attracted much attention in data mining research area. A fundamental problem formulation is the **sequential pattern** mining problem [2], which finds the complete set of frequent (closed) subsequences from an input sequence database SDB with a user-specified support threshold min_sup.

There exist several shortcomings of traditional sequential pattern mining which hinder its wide application. The first one is its huge result set. It is not uncommon that the complete set of sequential patterns contains millions of frequent subsequences. Although each frequent subsequence is statistically significant in terms of its support, it may not be interesting from an application point of view. Second, mining the complete set of frequent subsequences is inefficient, and is in fact impractical for the large and dense sequence databases when the minimum support threshold is low.

In this paper we explore the frequent subsequence-based clustering algorithm. Previous study has shown that frequent subsequences can be used as features for sequence clustering [5]. The approach proposed in [5] first adopts any existing sequential pattern mining algorithm to find the complete set of frequent subsequences from which a subset of subsequences can be further identified with special attention and used as features to project the input sequences into a new vector space. Finally it applies the traditional vector-space K-means clustering algorithm to find a given number of clusters. However, as analyzed above, this can be

inefficient when the database is large and dense. Differently from the previous approach, our basic idea here is to directly mine a subset of frequent subsequences which can be used as a condensed representation of the original database instead of being treated as features. For each input sequence, S_i, one subsequence supported by S_i is mined and used as a concise **summarization** of S_i. The input sequences with the same summarization subsequence can form a **micro-cluster**. The set of micro-clusters can be further grouped into a final set of **macro-cluster**s using any clustering paradigm (Note in our implementation we adopted the agglomerative hierarchical clustering algorithm with our own definition of the similarity between two micro-clusters based on their summarization subsequences). This two-step summarization subsequence-based clustering outperforms the traditional whole sequence-based clustering algorithm in two folds. On one hand, clustering on a concise data representation can lead to more efficient algorithm, on the other hand, by leaving out the noisy events from the input sequences, the summarization subsequence representation is also potentially helpful for improving the clustering accuracy.

$$(1.1) \qquad SIM(C_\lambda) = \frac{|s_\lambda| \times l}{\sum_{i=1}^{l} |S_i|}$$

Consider a summarization subsequence s_λ and its corresponding micro-cluster C_λ consisting of l input sequences, S_1, S_2, ..., S_l. As described above s_λ is a common subsequence among sequences in C_λ, we expect the summarization subsequence s_λ can be used to evaluate the internal similarity of micro-cluster C_λ, $SIM(C_\lambda)$, in a way as defined by Equation 1.1.

In Equation 1.1, $|s_\lambda|$ and $|S_i|$ denote the length of summarization subsequence s_λ and the length of input sequence S_i, respectively. We see that one way to maximize $SIM(C_\lambda)$ is to try to maximize the length of the summarization subsequence, that is, we hope the discovered summarization subsequence contains as many events as possible. However, Equation 1.1 is based on an unreal assumption, which means each distinct event in s_λ contributes equally to the internal similarity of micro-cluster C_λ. In fact, in many real cases the events in a sequence database may not be equally differentiable in terms of clustering. A good clustering criterion function should reflect the differentiability of each event. In the following we use a weight w_{e_i} ($0 < w_{e_i} \leq 1$) to denote the relative differentiability for a unique event e_i ($1 \leq i \leq n$). For simplicity, here we temporarily suppose all the events are equally differentiable (i.e., they have an equal weight), and let $w_{e_i}=1$ ($\forall i$, $1 \leq i \leq n$). In our real implementation, we assign a weight, $w_{e_i}=(\frac{1}{sup^{SDB}(e_i)})^\tau$, to any event e_i, where $sup^{SDB}(e_i)$ is the global support of event e_i, τ ($\tau \geq 0$) is called the **weight differentia factor** and by default τ is set to 1. In this way, an

Input Sequences	Summarization Subsequences
C A B A C	CBAC:2
A B A A	ABA:3
A B C B	ABCB:2
C B A C	CBAC:2
A C B B A	ABA:3, ABB:3, ACB:3
A B C B C	ABCB:2

Table 1: An example sequence database SDB.

event with higher global frequency is given a lower weight, and this kind of handling is similar to the Inverse Document Frequency measure in the TF*IDF term weighting scheme, which has been popularly adopted in information retrieval. The weight of a sequence, $S=e_1'e_2'...e_m'$, can then be computed as $W_S=\sum_{i=1}^{m} w_{e_i'}=\sum_{i=1}^{m}(\frac{1}{sup^{SDB}(e_i')})^\tau$. Given the concept of the event differentiability, the internal similarity of micro-cluster C_λ, $SIM(C_\lambda)$, can be defined in a better way as shown in Equation 1.2.

$$(1.2) \qquad SIM(C_\lambda) = \frac{W_{s_\lambda} \times l}{\sum_{i=1}^{l} W_{S_i}}$$

A good criterion to choose the summarization subsequence for an input sequence S_i is to try to maximize the internal similarity of the micro-cluster that S_i belongs to. For a micro-cluster C_λ, which contains a fixed set of input sequences, it may support multiple common subsequences, thus may have multiple choices for a summarization subsequence. In order to maximize the internal similarity of each micro-cluster as defined in Equation 1.2, the **summarization subsequence** s_λ of an input sequence S_i is heuristically defined as one of the frequent subsequences that are supported by S_i and have the largest weight. The Summarization Subsequence of input sequence S_i is denoted by SS_{S_i}. Note that given a minimum support *min_sup*, we may not be able to find any frequent covering subsequences for an input sequence. In this case we treat the input sequence as an outlier.

EXAMPLE 1. *The first column of Table 1 shows the input sequence database SDB in our running example. The database has a total of three unique items and six input sequences (i.e., $|SDB|$=6). Suppose min_sup = 2, the corresponding summarization subsequences supported by each input sequence are shown in column 2. We can see from Table 1 that the input sequence ACBBA has three summarization subsequences, ABA:3 (Note here '3' is the absolute support of ABA.), ABB:3, and ACB:3.* ■

The goal of this study is to mine one summarization subsequence for each non-outlier input sequence, and use the set of summarization subsequences to construct a sequence clustering algorithm. For this purpose, we devise an algorithm, CONTOUR [1], in order to perform this task. When an in-

---

[1]**CONTOUR** stands for efficiently mining the **CO**vering freque**NT** summarization subsequences f**O**r cl**U**ste**R**ing.

put sequence supports several summarization subsequences, CONTOUR only reports the one discovered first. For example, in Table 1, CONTOUR only reports ABA:3 as the summarization subsequence for input sequence ACBBA if ABA:3 is discovered before ABB:3 and ACB:3, and the set of summarization subsequences mined by CONTOUR in our running example is {CBAC:2, ABA:3, ABCB:2}.

The paper is organized as follows. Section 2 describes the CONTOUR algorithm in detail. Section 3 introduces a summarization subsequence-based sequence clustering algorithm. Section 4 presents the experimental results. Finally we will conclude with Section 5.

2 Efficiently Mining Summarization Subsequences

In this section, we describe a basic framework for frequent subsequence enumeration and introduce several optimization techniques to improve the performance of mining summarization subsequences. The CONTOUR algorithm can be built by integrating the optimization techniques with the subsequence enumeration framework.

2.1 Frequent Subsequence Enumeration

Mining the complete set of frequent subsequences from a given sequence database has been studied extensively in recent years, and various efficient mining methods have been proposed. In this paper, we choose the pattern growth paradigm [6, 7] as the general framework to mine the summarization subsequences. Under this framework, there always exists a current prefix subsequence before the algorithm finishes, which is initially set to empty. For each prefix, the mining algorithm builds its projected database, and computes the set of locally frequent events by scanning the projected database. Here the projected database consists of a set of projected sequences, where a **projected sequence** is defined as the subsequence of an input sequence after the first appearance of the prefix. If an input sequence does not contain the prefix, its corresponding projected sequence is empty. Each locally frequent event will be chosen according to a certain ordering scheme such as lexicographic ordering and used to extend the current prefix to get a new prefix. The same procedure will be applied to the new prefix in a recursive way. This mining process complies with depth-first traversal and the divide-and-conquer paradigm.

Note that there are two popular methods in constructing the projected databases. One is the so called *physical projection*, another is *pseudo projection* [6]. *Pseudo projection* uses a pointer to record the starting place of a projected sequence, thus avoids building and maintaining the physically projected sequences. It is a common belief that *pseudo projection* is more space efficient than *physical projection*. CONTOUR adopts the *pseudo projection* to build projected database. We refer the interested readers to [6, 7] for more details about the *pseudo projection* method.

In the above discussion we mentioned that the lexicographical ordering can be used to sort the locally frequent events. However, because our goal is to find a set of high-quality summarization subsequences which are good for clustering, the lexicographical ordering may not be the best ordering scheme from the clustering point of view. Among other event ordering schemes, support ascending ordering and support descending ordering are two candidates which are popular in pattern discovery. As an input sequence may support multiple summarization subsequences, CONTOUR prefers the summarization subsequences with low support. We made this decision based on the heuristic that the summarization subsequences with very high support are usually not very differentiable in terms of class labels. Thus, in CONTOUR the ascending ordering by support is adopted as the default ordering scheme.

2.2 Optimization Techniques for Summarization Subsequence Discovery

A naïve method to mine the set of summarization subsequences is that in the above frequent subsequence enumeration process we always maintain for each input sequence a frequent subsequence S_i that has the largest weight and was discovered first so far, and when the algorithm finishes, each currently maintained frequent subsequence becomes a summarization subsequence. As the set of summarization subsequences is only a subset of the set of all frequent subsequences, this enables us to devise some effective pruning methods to prune the unpromising search subspaces that do not contain any summarization subsequence.

2.2.1 Closed Sequence-based Optimization

The first class of optimization techniques that can be exploited by CONTOUR is based on the following property of a summarization subsequence.

PROPERTY 2.1. *A summarization subsequence must be a closed subsequence, but may not be a maximal subsequence* [2].

Proof. *We will prove the first part by contradiction. Suppose a summarization subsequence, p, is not closed, there must exist one of its super-sequences, p', such that $p \sqsubseteq p'$ and $sup^{SDB}(p) = sup^{SDB}(p')$. This means $W_{p'} > W_p$, which contradicts with the definition of a summarization subsequence. The second part can be easily proved by a counterexample shown in Table 1. As shown in Table 1 ABB:3 is a summarization subsequence, but it is not maximal, as one of its super-sequence, ABCB:2, is frequent and in fact it is also a summarization subsequence.*∎

_____
[2]A subsequence p is a *closed* subsequence if none of its super-sequences has the same support as p, while p is a *maximal* subsequence if none of its super-sequences is frequent.

Property 2.1 implies that some optimization techniques proposed for closed sequence mining can be exploited to mine summarization subsequences. In CONTOUR, we applied the **BackScan** search space pruning method to enhance the algorithm efficiency. Due to limited space, we will not introduce it in detail here but give a simple example as an illustration, and refer the interested readers to [7] for an indepth description of the method. In our running example as shown in Table 1, suppose the current prefix sequence is BAC:2, which appears in the first and the fourth input sequences. It can be seen that there exists an event 'C' that always occurs before subsequence BAC in both the first and the fourth input sequences, in this case, we do not need to mine the frequent subsequences with prefix BAC:2, which can be safely pruned.

2.2.2 Unpromising Projected Sequence Pruning

Although Property 2.1 illustrates that a summarization subsequence must be a closed subsequence, CONTOUR does not need to perform the sequence closure checking in order to ensure each mined summarization subsequence is a closed one, as long as we can make sure that a summarization subsequence w.r.t. an input sequence S_i, SS_{S_i}, is one of the covering frequent subsequences of S_i that have the largest weight. Because a frequent closed subsequence may not be among the frequent covering subsequences that have the largest weight for any input sequence, the set of summarization subsequences is just a subset of all frequent closed subsequences, thus, we should also devise some methods to prune the unpromising search subspaces that do not contain any summarization subsequence.

In the following, we use $CFCS_{S_i}$ to denote the Current Frequent Covering Subsequence w.r.t. an input sequence S_i that has the largest weight and was discovered first so far. Let the current prefix subsequence be p, its support be $sup^{SDB}(p)$, its projected database (i.e., the set of projected subsequences w.r.t. p) be $SDB|_p=\{PS_1, PS_2, ..., PS_{sup^{SDB}(p)}\}$. By scanning $SDB|_p$ once, the locally frequent events in $SDB|_p$ can be found, and the locally infrequent events are removed from $SDB|_p$. The projected database w.r.t. prefix p without infrequent events is denoted by $SDB|'_p=\{PS'_1, PS'_2, ..., PS'_{sup^{SDB}(p)}\}$. Note that any projected subsequence in $SDB|'_p$ (or $SDB|_p$) can be empty. An important observation is that some short projected sequences may not contain sufficient number of events to generate any summarization subsequence. Thus, they should be identified and pruned[3].

DEFINITION 1. *(Trivial projected sequence) A projected sequence PS'_i in $SDB|'_p$ (where $1 \leq i \leq sup^{SDB}(p)$), is a **triv-***

---

[3]In essence, it shares a similar spirit with the pruning methods adopted in [8, 9].

ial *projected sequence if it satisfies one of the following two conditions (Note that in Equation 2.4, $j \in [1..sup^{SDB}(p)]$):*

$$(2.3) \qquad (W_{PS'_i} + W_p) \leq W_{CFCS_{S_i}}$$

$$(2.4) \quad |\{ \forall j, PS'_j | (W_{PS'_j} + W_p) > W_{CFCS_{S_i}} \}| < min\_sup$$

*Otherwise, PS'_i is said to be **non-trivial**.* ∎

For any projected sequence PS'_i w.r.t. prefix p, the upper bound of the weight of a frequent subsequence grown from p and supported by PS'_i is $(W_{PS'_i}+W_p)$. The first case in Definition 1 (i.e., Equation 2.3) states that its upper bound is no greater than the weight of the currently maintained frequent covering subsequence of S_i (Here we use S_i to denote the corresponding input sequence of PS'_i). Thus, there is no subsequence derived by extending prefix p and supported by PS'_i, and also has a weight larger than $W_{CFCS_{S_i}}$. In the second case of Definition 1 (i.e., Equation 2.4), it states that although the weight of projected sequence PS'_i may be large enough to derive subsequences whose weight is greater than $W_{CFCS_{S_i}}$, the number of projected sequences with large weight in $SDB|'_p$ is not sufficient to obtain a subsequence whose weight is larger than $W_{CFCS_{S_i}}$ and is also frequent.

Although a trivial projected sequence PS'_i cannot be used to derive any summarization subsequence for input sequence S_i, it may contribute to some summarization subsequences grown from other non-trivial projected sequences. Thus, such sequences should not be simply pruned. However, a trivial projected sequence PS'_i satisfying Equation 2.5 can be safely pruned according to the following Lemma.

LEMMA 2.1. *(Trivial projected sequence pruning) A trivial projected sequence w.r.t. prefix p and input sequence S_i, PS'_i, cannot be used to derive any summarization subsequence by extending prefix p, if the following condition holds (where $j \in [1, sup^{SDB}(p)]$):*

$$(2.5) \quad (W_{PS'_i} + W_p) \leq \min_{\forall j,\ PS'_j\ is\ non-trivial} W_{CFCS_{S_j}}$$

Proof. *We prove it by contradiction. Suppose PS'_i contributes to the derivation of a summarization subsequence, SS_{S_j}, by extending prefix p, whose weight is greater than $W_{CFCS_{S_j}}$ ($\forall j$, PS'_j is non-trivial). Because the upper bound of the weight of any subsequence extended from p and supported by PS'_i is $(W_{PS'_i} + W_p)$, we have:*

$$(W_{PS'_i} + W_p) \geq W_{SS_{S_j}}$$

Also, the following equation holds:

$$W_{SS_{S_j}} > W_{CFCS_{S_j}}$$

Thus, we get:

$$(W_{PS'_i} + W_p) > W_{CFCS_{S_j}}$$

which contradicts with Equation 2.5. ∎

Lemma 2.1 introduces a method to identify some un-promising projected sequences that can be safely pruned. In some cases, the projected database may not contain any non-trivial projected sequences, the entire projected database can then be safely pruned.

2.3 The CONTOUR Algorithm

By incorporating the optimization techniques described in Sections 2.2.1 and 2.2.2 into the sequence enumeration framework introduced in Section 2.1, we get the integrated CONTOUR algorithm. Due to limited space, we will not elaborate on it.

3 Summarization Subsequence based Clustering

After the set of summarization subsequences have been discovered by CONTOUR, they will be used to cluster the input sequences. We denote the Summarization Subsequence based Clustering by SSC. SSC is performed in two steps. First, a set of small micro-clusters are generated according to the discovered summarization subsequences. Next, a set of final macro-clusters are created by merging the micro-clusters generated in the first step. This two-step clustering paradigm is in essence very similar to the one adopted by several previous studies [10, 1].

3.1 Micro-cluster Generation

Once we have discovered the set of summarization subsequences, it is straightforward to generate the set of micro-clusters. In SSC the input sequences with the same summarization subsequence are grouped together to form a micro-cluster. As a summarization subsequence w.r.t. an input sequence S_i is defined as one of the frequent covering subsequences of S_i that have the largest weight, the internal similarity defined in Equation 1.2 of the corresponding micro-cluster can be approximately maximized. In SSC we use the prefix-tree structure to group the input sequences with the same summarization subsequence together.

3.2 Macro-cluster Creation

The number of micro-clusters is usually larger than the number of real clusters, thus we need to apply certain clustering algorithm to create K macro-clusters from the set of micro-clusters, where K is a user-specified parameter which indicates the real number of clusters in the input sequence database. In SSC, we adopt the agglomerative hierarchical clustering paradigm to create K macro-clusters. Initially each micro-cluster is treated as a macro-cluster, and if the number of macro-clusters is larger than K, two closest macro-clusters are merged to form a large macro-cluster, and this process is repeated until exact K macro-clusters are retained.

To apply the agglomerative hierarchical clustering paradigm, we first compute the similarity matrix of the set of micro-clusters. Suppose the number of micro-clusters generated by CONTOUR is N_{mi}, we use $\mathtt{MI}_j$ ($1 \leq j \leq N_{mi}$) to denote a micro-cluster, and $\mathtt{SS}_j$ to denote its corresponding summarization subsequence.

Before we define the similarity between any two micro-clusters, let us first define the similarity between any two sequences, S_1 and S_2. If we use $s_c = e_1 e_2 ... e_q$ to denote a common subsequence of S_1 and S_2, the similarity between S_1 and S_2, $SIM(S_1, S_2)$, is defined as the maximum sequence weight among all the common subsequences divided by the sum of the sequence weights of S_1 and S_2, which is shown in the following equation.

$$(3.6) \qquad SIM(S_1, S_2) = \frac{\max_{\forall s_c} W_{s_c}}{W_{S_1} + W_{S_2}}$$

The maximum sequence weight among all the common subsequences of S_1 and S_2, $\max_{\forall s_c} W_{s_c}$, can be computed using Dynamic Programming with time complexity of $O(|S_1| \times |S_2|)$, where $|S_1|$ and $|S_2|$ are the lengths of S_1 and S_2 respectively [3]. The similarity between any two micro-clusters, $\mathtt{MI}_1$ and $\mathtt{MI}_2$ can be defined to be the similarity between their corresponding summarization subsequences, $\mathtt{SS}_1$ and $\mathtt{SS}_2$, that is,

$$(3.7) \qquad SIM(\mathtt{MI}_1, \mathtt{MI}_2) = SIM(\mathtt{SS}_1, \mathtt{SS}_2)$$

After defining the similarity between two micro-clusters, we now turn to define the *Group Average* similarity between two macro-clusters, $\mathtt{MA}_1$ and $\mathtt{MA}_2$. If we use N_{MA_1} and N_{MA_2} to denote the number of micro-clusters in $\mathtt{MA}_1$ and $\mathtt{MA}_2$ respectively, and $\mathtt{MI}_i^1$ ($1 \leq i \leq N_{MA_1}$) and $\mathtt{MI}_j^2$ ($1 \leq j \leq N_{MA_2}$) to denote a micro-cluster in $\mathtt{MA}_1$ and $\mathtt{MA}_2$ respectively, the *Group Average* similarity between $\mathtt{MA}_1$ and $\mathtt{MA}_2$ is defined as follows.

$$(3.8)$$
$$SIM(\mathtt{MA}_1, \mathtt{MA}_2) = \frac{\sum_{i=1}^{N_{MA_1}} \sum_{j=1}^{N_{MA_2}} SIM(\mathtt{MI}_i^1, \mathtt{MI}_j^2)}{N_{MA_1} \times N_{MA_2}}$$

As we initially treat each micro-cluster as a macro-cluster, the initial similarity matrix can be very easily computed according to Equation 3.7. At each stage of the agglomerative hierarchical clustering, the SSC algorithm needs to update the similarity matrix upon merging two closest macro-clusters and getting a new macro-cluster. Let the newly generated macro-cluster be $\mathtt{MA}_1$ and the two component macro-clusters forming $\mathtt{MA}_1$ be $\mathtt{MA}_{11}$ and $\mathtt{MA}_{12}$, and

denote the number of micro-clusters in $\texttt{MA}_{11}$ and $\texttt{MA}_{12}$ by $N_{MA_{11}}$ and $N_{MA_{12}}$ respectively. A naïve way to compute the similarity between $\texttt{MA}_1$ and another existing macro-cluster $\texttt{MA}_2$ can be based on Equation 3.8. However, this is inefficient. As we know, at the current stage SSC already maintains $SIM(\texttt{MA}_{11}, \texttt{MA}_2)$ and $SIM(\texttt{MA}_{12}, \texttt{MA}_2)$, a more clever way to compute $SIM(\texttt{MA}_1, \texttt{MA}_2)$ could be based on these already known information, as shown in Equation 3.9.

$$SIM(\texttt{MA}_1, \texttt{MA}_2) = \frac{\sum_{i=1}^{N_{MA_1}} \sum_{j=1}^{N_{MA_2}} SIM(\texttt{MI}_i^1, \texttt{MI}_j^2)}{N_{MA_1} \times N_{MA_2}} =$$

$$\frac{\sum_{k=1}^{N_{MA_{11}}} \sum_{j=1}^{N_{MA_2}} SIM(\texttt{MI}_k^1, \texttt{MI}_j^2)}{N_{MA_1} \times N_{MA_2}} +$$

$$\frac{\sum_{l=N_{MA_{11}}+1}^{N_{MA_1}} \sum_{j=1}^{N_{MA_2}} SIM(\texttt{MI}_l^1, \texttt{MI}_j^2)}{N_{MA_1} \times N_{MA_2}} =$$

$$\frac{SIM(\texttt{MA}_{11}, \texttt{MA}_2) \times N_{MA_{11}} + SIM(\texttt{MA}_{12}, \texttt{MA}_2) \times N_{MA_{12}}}{N_{MA_{11}} + N_{MA_{12}}}$$

(3.9)

One of the most time-consuming operations in agglomerative hierarchical clustering is to find the two closest macro-clusters which have the maximum similarity among all pairs of macro-clusters. In CONTOUR, the similarity matrix is indexed by a red-black tree structure where the similarity is designated as the key and updated synchronously with the similarity matrix. As a red-black tree with n internal nodes has height at most $2\lg(n+1)$ [3], the search of the maximum similarity can be efficiently implemented in $O(\lg n)$, where n is the number of pairs of initial micro-clusters. We should note that the number of initial micro-clusters is usually much smaller than the number of sequences in the input database.

4 Empirical Results

Our performance study shows that CONTOUR is very efficient and the pruning techniques proposed in this paper are effective in improving the algorithm efficiency. Figure 1a) shows the comparison result with BIDE, while Figure 1b) evaluates the effectiveness of the pruning techniques with dataset *Snake*.

We also conducted an extensive performance study to evaluate the scalability of CONTOUR, and the accuracy of the frequent subsequence-based clustering algorithm, SSC. The results show that CONTOUR has good scalability, and SSC is a promising approach for clustering sequentialized XML documents and achieves better clustering quality than the latest structure summary based XML clustering algorithm [4].
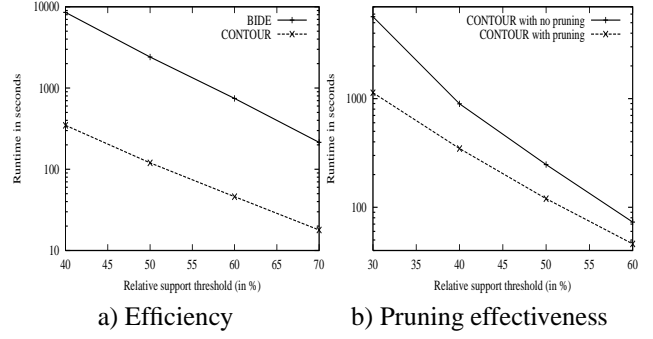

a) Efficiency   b) Pruning effectiveness

Figure 1: Efficiency and pruning effectiveness test (*Snake*)

5 Conclusions

In this paper we devise a simple and efficient algorithm, CONTOUR, to mine a set of summarization subsequences, which is a concise representation of the original sequence database and preserves much structural information, and can be used to efficiently cluster the input sequences with a high clustering quality. Our performance study shows that CONTOUR is efficient to mine the set of summarization subsequences, and the optimization techniques are effective in improving the efficiency of CONTOUR. In addition, the summarization subsequence based clustering algorithm, SSC, can generate high quality clustering results, and provides a promising alternative approach to sequence clustering.

References

[1] C.C. Aggarwal, J. Han, J. Wang, P.S. Yu. *A Framework for Clustering Evolving Data Streams.* VLDB'03.

[2] R. Agrawal, R. Srikant. *Mining Sequential Patterns.* ICDE'95.

[3] T. Cormen, C. Leiserson, R. Rivest, C. Stein. *Introduction to Algorithms.* MIT Press, 2001.

[4] T. Dalamagas, T. Cheng, K. Winkel, T. Sellis. A Methodology for Clustering XML Documents by Structure. *Information Systems*, 31 (3):187-228, 2006.

[5] V. Guralnik, G. Karypis. *A scalable algorithm for clustering sequential data.* ICDM'01.

[6] J. Pei, J. Han, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.C. Hsu. *PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth.* ICDE'01.

[7] J. Wang, J. Han. *BIDE: Efficient Mining of Frequent Closed Sequences.* ICDE'04.

[8] J. Wang, G. Karypis. *SUMMARY: Efficiently Summarizing Transactions for Clustering.* ICDM'04.

[9] J. Wang, G. Karypis. *HARMONY: Efficiently Mining the Best Rules for Classification.* SDM'05.

[10] T. Zhang, R. Ramakrishnan, M. Livny. *BIRCH: An Efficient Data Clustering Method for Very Large Databases.* SIGMOD'96.