Pattern-Oriented Hierarchical Clustering*

Tadeusz Morzy, Marek Wojciechowski, Maciej Zakrzewicz

Poznan University of Technology
Institute of Computing Science
ul. Piotrowo 3a, 60-965 Poznan, Poland
Tadeusz.Morzy@put.poznan.pl
Marek.Wojciechowski@cs.put.poznan.pl
Maciej.Zakrzewicz@cs.put.poznan.pl

Abstract. Clustering is a data mining method, which consists in discovering interesting data distributions in very large databases. The applications of clustering cover customer segmentation, catalog design, store layout, stock market segmentation, etc. In this paper, we consider the problem of discovering similarity-based clusters in a large database of event sequences. We introduce a hierarchical algorithm that uses sequential patterns found in the database to efficiently generate both the clustering model and data clusters. The algorithm iteratively merges smaller, similar clusters into bigger ones until the requested number of clusters is reached. In the absence of a well-defined metric space, we propose the similarity measure, which is used in cluster merging. The advantage of the proposed measure is that no additional access to the source database is needed to evaluate the inter-cluster similarities.

1 Introduction

Clustering is one of the most popular data mining methods [2] [3] [4] [5] [6] [7] [8] [9] [11]. It consists in discovering interesting data distributions and patterns in very large databases. Given *k* data points in a *d*-dimensional metric space, the problem of clustering is to partition the data points into *n* clusters such that the data points within a cluster are closer (more similar) to each other than data points in different clusters. Clustering is often used for market segmentation, in which the customers are divided into groups based on the similarity of their characteristics.

Clustering algorithms typically determine *n* partitions that optimize some criterion function. The most commonly used criterion is the square-error criterion defined as follows:

$$E = \sum_{i=1}^{n} \sum_{p \in c_i} \left\| p - m_i \right\|^2 \qquad (1)$$

* This work was partially supported by the grant no. KBN 43-1309 from the State Committee for Scientific Research (KBN), Poland.

where m_i is the mean of cluster c_i, p is a data point, and E is the square error. Many clustering algorithms employ the idea of *hierarchical clustering*, which consists in merging pairs of similar clusters to form new larger clusters.

Traditional clustering approaches deal with points in d-dimensional space. However, not all types of information can be represented in this form. In many applications, users operate on databases of event sequences, such as customer purchase history given in Figure 1, where an ordered set of purchased products is stored for each customer. Let us consider the general problem of the similarity of sequences. It seems that e.g. the sequences $a \rightarrow b \rightarrow c$ and $b \rightarrow c \rightarrow d$ are similar since the both contain the same subsequence $b \rightarrow c$. But what can we say about the similarity of the sequences e.g. $a \rightarrow b$ and $c \rightarrow d$? We claim that these two sequences also can be considered similar, if there are many other sequences in the database, that contain them both, e.g. $a \rightarrow b \rightarrow c \rightarrow d$, $a \rightarrow c \rightarrow b \rightarrow d$, etc. Therefore, we assume that two sequences are similar if either they contain the identical subsequences, or their subsequences have the tendency to co-occur together in some other sequences. For example, the customer *103* is more similar to the customer *104* than to the customer *105* since the pair *103-104* has a common subsequence (*tv_set* $\rightarrow$ *vcr* $\rightarrow$ *cassette*) while the pair *103-105* has no common subsequences. We are interested in clustering sequences based on such intuitive similarity measure. We notice that this problem cannot be solved using traditional clustering methods because: 1. the sequences are variable-length, 2. the sequences cannot be represented in a d-dimensional metric space, and 3. no natural distance function is available.

In this paper we address and solve the problem of partial clustering of sequential data. We are interested in discovering an arbitrary number of possibly overlapping clusters that hold the customers, whose behavior is similar to each other. We refer to our clustering method as to *partial* clustering, because we allow the customers who are not similar to any other not to be covered by any cluster, and we allow a customer to belong to more than one cluster. To perform the partial clustering of sequential data, we employ the idea of sequential pattern discovery [1] [10]. A sequential pattern is a frequently occurring subsequence of database sequences. Sequential pattern discovery consists in finding all sequential patterns, whose frequency is above some user-defined minimum value. Thus, the discovered sequential patterns represent the most common subsequences within the database sequences and can be used to determine their similarity. For example, the sequential patterns that can be discovered in the database from Figure 1 are: '*tv_set* $\rightarrow$ *vcr* $\rightarrow$ *cassette*' (contained in two database sequences), *'book* $\rightarrow$ *c_disk'* (also contained in two), etc. Each of those sequential patterns says that a number of customers bought one product, later on they bought some other product, and so on.

The presented algorithm uses sequential patterns discovered in the database to generate both the clustering model and cluster contents. For example, our algorithm executed on the database from Fig. 1 with $n=2$ gives two clusters based on the following clustering model:

- Cluster 1: described by the patterns: *tv_set* $\rightarrow$ *vcr* $\rightarrow$ *cassette* and *bicycle* $\rightarrow$ *b_ball*; the cluster contains the following customers: 102, 103, 104,
- Cluster 2: described by the patterns: *book* $\rightarrow$ *c_disk* and *lamp* $\rightarrow$ *pillow*; the cluster contains the following customers: 101, 105.

cust_id	sequence
101	lamp → l_bulb → pillow → book → c_disk
102	t_rocket → bicycle → b_ball → s_bindings
103	tv_set → vcr → c_phone → cassette
104	bicycle → tv_set → b_ball → vcr → cassette
105	book → c_disk → dryer → lamp → pillow → d_washer

Fig. 1. Example of customer purchase history database

1.1 Related Work

In recent years, a number of clustering algorithms for large databases has been proposed. In [8], a clustering method based on randomized search, called CLARANS has been introduced. CLARANS was dedicated to solve problems of data mining in spatial databases. The problem of clustering in large spatial databases was also addressed in [2] and [3]. In [11], the authors presented a clustering method named BIRCH whose I/O complexity was a little more than one scan of the data. In [5], the hierarchical clustering algorithm named CURE was presented. The algorithm was designed for identifying clusters having non-spherical shapes. In [6] and [7], a method for hypergraph-based clustering of data in a high dimensional space has been presented. In [9], a clustering method for data without distance functions was considered, and the proposed algorithm tried to group together records that had frequently co-occurring items. The implementation of the algorithm used frequent item sets discovered by the association rule algorithm as hypergraph edges. [4] described a novel approach for clustering collections of sets, and its application to the analysis and mining of categorical data. The proposed algorithm facilitated a type of similarity measure arising from the co-occurrence of values in the dataset. Unfortunately, none of the works addressed the problem of clustering of sequences of events.

The problem of frequent pattern discovery in sequential data was introduced in [1] and three different algorithms were proposed. In [10], the problem was generalized by adding time constraints and taxonomies.

1.2 Paper Outline

The paper is organized as follows. In Section 2, the basic definitions and the formulation of the problem are given. Section 3 contains the problem decomposition and the description of the algorithm for pattern-oriented clustering. The idea behind our algorithm is illustrated by a detailed example. We conclude with a summary and directions for future work in Section 4.

2 Problem Formulation

2.1 Definitions

Let $L = \{l_1, l_2, ..., l_m\}$ be a set of literals called items. A *sequence* $S = <X_1 X_2 ... X_n>$ is an ordered list of sets of items such that each set of items $X_i \subseteq L$. Let the database D be a set of sequences.

We say that the sequence $S_1 = <Y_1 Y_2 ... Y_m>$ *supports* the sequence $S_2 = <X_1 X_2 ... X_n>$ if there exist integers $i_1 < i_2 < ... < i_n$ such that $X_1 \subseteq Y_{i1}$, $X_2 \subseteq Y_{i2}$, ..., $X_n \subseteq Y_{in}$. We also say that the sequence S_2 is a *subsequence* of the sequence S_1 (denoted by $S_2 \subset S_1$).

A *frequent pattern* is a sequence that is supported by more than a user-defined minimum number of sequences in D. Let P be a set of all frequent patterns in D.

A *cluster* c is an ordered pair $<Q,S>$, where $Q \subseteq P$ and $S \subseteq D$, and S is a set of all database sequences supporting at least one pattern from Q. We call Q a *cluster description*, and S a *cluster content*. We use a dot notation to refer to a cluster description as to $c.Q$ and to a cluster content as to $c.S$.

A *union* c_{ab} of the two clusters c_a and c_b is defined as follows:

$c_{ab} = union(c_a, c_b) = < c_a.Q \cup c_b.Q , c_a.S \cup c_a.S >$

Inter-cluster similarity is a co-occurrence function $f(c_1, c_2)$. In this paper, we use the following co-occurrence function:

$$f(C_1, C_2) = \frac{|C_1.S \cap C_2.S|}{|C_1.S \cup C_2.S|} \quad . \qquad (2)$$

The above similarity function returns values from the range of $<0;1>$, where the value of *1* means that the clusters are identical while the value of *0* means that the clusters exhibit no similarity at all.

2.2 Problem statement

Given a database $D = \{s_1, s_2, ..., s_n\}$ of data sequences, and a set $P = \{p_1, p_2, ..., p_m\}$ of frequent patterns in D, the problem is to divide P into a set of clusters, such that $\forall_{i,j,i \neq j} C_i.Q \cap C_j.Q = \varnothing$, and inter-cluster similarity is minimized.

3 Pattern-Oriented Hierarchical Clustering

In this section, we describe a new clustering algorithm POPC for clustering large volumes of sequential data. The algorithm implements the general idea of hierarchical clustering. However, instead of starting with a set of clusters containing one data sequence each, our algorithm uses previously discovered frequent patterns and starts with clusters containing data sequences supporting the same frequent pattern. We assume that a well-known algorithm for frequent pattern discovery is executed before.

3.1 Algorithm POPC

The algorithm for partial clustering based on frequently occurring patterns is decomposed into the following phases:

- *Transformation Phase*, which prepares the database for effective similarity evaluations,
- *Merge Phase*, which iteratively reduces the number of clusters by merging the most similar ones,
- an optional *Description Pruning Phase*, which can be applied to compress the generated clustering model.

3.1.1 Transformation Phase

In this phase, the database is transformed into a pattern-oriented form, which is more suitable for evaluating unions and intersections of cluster contents (used in the subsequent phases). For each frequent pattern we keep an ordered list of data sequences supporting the pattern. Each data sequence is represented by its identifier, e.g. in our example of sequences corresponding to lists of products bought by clients of a supermarket, a sequence could be identified by a unique identifier of a client. Sequences that do not support any frequent pattern are ignored.

Each pattern, together with the list of sequences supporting it, constitutes a cluster whose description is a set that contains the pattern as its only element. The cluster's content is made up of a set of data sequences from the list.

The proposed database representation simplifies evaluation of inter-cluster similarities. There is no need to refer to the original database in subsequent phases of the algorithm. Moreover, the size of the transformed database reduces as clusters are being merged together. When the process is finished, the database contains the result of clustering (descriptions and contents of the discovered clusters).

3.1.2 Merge Phase

Figure 2 presents the Merge Phase of the clustering algorithm. First, the m patterns are mapped into m clusters, forming an initial set of clusters C_1, where each cluster is described by exactly one pattern. In the next step, the similarity function values are evaluated for all possible combinations of clusters. The similarity values are stored in a form of a matrix M_1. Next, the algorithm iteratively merges together pairs of clusters according to their similarity values and cluster contents' sizes. In each iteration k, the two most similar clusters $c_a, c_b \in C_k$ are determined, and replaced by a new cluster $c_{ab} = union(c_a, c_b)$. If there are several pairs of clusters having maximal similarity values, then the two clusters having the smallest contents are merged. The actual merging is done by the function called *cluster*, described in detail in Section 3.1.2.2. When the new cluster is created, the matrix containing similarity values has to be re-evaluated. This operation is performed by means of the function called *simeval*, described in Section 3.1.2.1.

The Merge Phase stops when the number of clusters reaches n or when there is no such pair of clusters $c_a, c_b \in C_k$ whose similarity is greater than 0. The latter condition implies that the algorithm may discover a larger number of clusters than requested by a user. In this case, the number of discovered clusters (as well as the fraction of the

original database covered by them) depends on the number and strength of frequent patterns used for clustering. If the quality of clustering is unsatisfactory, the clustering should be repeated with a higher number of frequent patterns (a set of patterns satisfying a lower frequency threshold).

$C_1 = \{c_i: c_i.Q=\{p_i\}, c_i.S=\{s_j: s_j \in D \wedge s_j \textit{ supports } p_i\}\}$;
$M_1 = \textit{simeval}(C_1, \varnothing)$;
k=1;
while $|C_k| > n$ and exist $c_a, c_b \in C_k$ such that $f(c_a,c_b) > 0$ ***do begin***
  $C_{k+1} = \textit{cluster}(C_k, M_k)$;
  $M_{k+1} = \textit{simeval}(C_{k+1}, M_k)$;
  k++;
end;
Answer $=C_k$;

Fig. 2. Merge Phase

3.1.2.1 Similarity Matrix Evaluation: simeval

Similarity matrix M_l stores the values of the similarity function for all possible pairs of clusters in an *l*-th algorithm iteration. The cell $M_l(x,y)$ represents the similarity value for the clusters c_x and c_y from the cluster set C_l (see example in Figure 3). The function *simeval* computes the values of the similarity matrix M_{l+1}, using both the similarity matrix M_l and the current cluster contents. Notice that in all iterations except the first one, the similarity matrix need not be completely re-computed. Only the similarity values concerning the newly created cluster have to be evaluated. Due to diagonal symmetry of the similarity matrix, for *k* clusters, only $(k^2-k)/2$ similarity function values need to be computed before the first iteration, and only $(k-1)$ in the subsequent ones.

In each iteration, the size of the matrix decreases since two rows and two columns corresponding to the clusters merged to form a new one are removed and only one column and one row are added for a newly created cluster.

-	$f(c_2, c_1)$	$f(c_3, c_1)$
$f(c_1,c_2)$	-	$f(c_3, c_2)$
$f(c_1,c_3)$	$f(c_2, c_3)$	-

$f(c_1,c_2) = f(c_2, c_1)$
$f(c_1,c_3) = f(c_3, c_1)$
$f(c_2, c_3) = f(c_3, c_2)$

Fig. 3. Structure of the similarity matrix for three clusters

3.1.2.2 Cluster Merging: cluster

In each iteration, the number of processed clusters decreases by one. The similarity-based merging is done by the function called *cluster*. The function *cluster* scans the similarity matrix and finds pairs of clusters, such that their similarity is maximal. If there are many pairs of clusters that reach the maximal similarity values, then the function *cluster* selects the one with the smallest size of the union of their contents. Notice that no access to the original database is required to perform this phase of the

algorithm. The function *cluster* takes a set of clusters C_k as one of its parameters and returns a set of clusters C_{k+1} such that $C_{k+1} = (C_k \setminus \{c_a, c_b\}) \cup \{c_{ab}\}$, where $c_a, c_b \in C_k$ are clusters chosen for merging and $c_{ab} = union(c_a, c_b)$.

3.1.3 Description Pruning Phase (optional)

The Merge Phase returns the complete set of the requested clusters. However, the clusters may have descriptions that are not minimal. That is, some patterns included in a description might be redundant in such a way that a set of data sequences supporting the pattern will always be a subset of data sequences supporting some other pattern included in the same description. In the optional Description Pruning Phase, descriptions of all clusters can be tested whether they include a pair of patterns p_a, p_b such that $p_a \subset p_b$. If such a pair is found, p_b is removed from the description. Figure 4 presents the Description Pruning Phase of the algorithm.

```
C = the set of clusters generated in the Merge Phase;
for each c_i ∈ C do
    while exist p_a,p_b ∈ c_i.Q such that p_a ⊂ p_b do
        c_i.Q = c_i.Q \ { p_b };
```

Fig. 4. Description Pruning Phase

Notice that it is also possible to perform pruning of descriptions within the Merge Phase for each newly created cluster.

3.2 Example

Consider a database of customer transactions shown in Figure 5. For each transaction, we keep the transaction's time, items bought in the transaction and a unique customer identifier. Figure 6 shows an alternative representation of the database, where an ordered set of purchased items is given for each customer.

Let us assume that a user wants to cluster customers who follow similar frequent buying patterns into three clusters. Figure 7 shows frequent sequential patterns discovered in the database from Figure 5 (with a support threshold of 25%). The clustering algorithm starts with the Transformation Phase, which results in the initial set of clusters shown in Figure 8.

Customer Id	Transaction Time	Items Bought
1	October 10 1998	10 60
1	December 10 1998	20 30
1	December 15 1998	40
1	February 19 1999	50
2	November 10 1998	40
2	November 21 1998	50
2	December 12 1998	10
2	January 18 1999	20 30 70
3	October 15 1998	40
3	November 29 1998	50
3	December 14 1998	10
3	January 22 1999	80
3	February 11 1999	20 30
4	December 20 1998	10
4	February 4 1999	20
5	February 12 1999	80
6	November 1 1998	10
6	November 22 1998	30 90
7	February 1 1999	20 30
8	October 10 1998	60
8	November 22 1998	100
9	January 12 1999	100
10	January 21 1999	90 100

Fig. 5. Database sorted by Customer ID and Transaction Time

ID	Customer sequence
1	< (10 60) (20 30) (40) (50) >
2	< (40) (50) (10) (20 30 70) >
3	< (40) (50) (10) (80) (20 30) >
4	< (10) (20) >
5	< (80) >
6	< (10) (30 90) >
7	< (20) (30) >
8	< (60) (100) >
9	< (100) >
10	< (90 100) >

Fig. 6. Customer-sequence representation of the database

Patterns with support > 25%	
p_1	< (10) (20 30) >
p_2	< (10) (20) >
p_3	< (10) (30) >
p_4	< (20 30) >
p_5	< (10) >
p_6	< (20) >
p_7	< (30) >
p_8	< (40) (50) >
p_9	< (40) >
p_{10}	< (50) >
p_{11}	< (100) >

Fig. 7. Pattern set used for clustering

Cluster	Description	Sequences
c_a	p_1	1, 2, 3
c_b	p_2	1, 2, 3, 4
c_c	p_3	1, 2, 3, 6
c_d	p_4	1, 2, 3, 7
c_e	p_5	1, 2, 3, 4, 6
c_f	p_6	1, 2, 3, 4, 7
c_g	p_7	1, 2, 3, 6, 7
c_h	p_8	1, 2, 3
c_i	p_9	1, 2, 3
c_j	p_{10}	1, 2, 3
c_k	p_{11}	8, 9, 10

Fig. 8. Pattern-oriented representation of the database

Before the first iteration of the Merge Phase the similarity matrix has to be build. The similarity matrix for the initial set of clusters from Figure 8 is shown in Figure 9.

	c_a	c_b	c_c	c_d	c_e	c_f	c_g	c_h	c_i	c_j	c_k
c_a	x	0.75	0.75	0.75	0.6	0.6	0.6	1	1	1	0
c_b	0.75	x	0.6	0.6	0.8	0.8	0.5	0.75	0.75	0.75	0
c_c	0.75	0.6	x	0.6	0.8	0.5	0.8	0.75	0.75	0.75	0
c_d	0.75	0.6	0.6	x	0.5	0.8	0.8	0.75	0.75	0.75	0
c_e	0.6	0.8	0.8	0.5	x	0.66	0.66	0.6	0.6	0.6	0
c_f	0.6	0.8	0.5	0.8	0.66	x	0.66	0.6	0.6	0.6	0
c_g	0.6	0.5	0.8	0.8	0.66	0.66	x	0.6	0.6	0.6	0
c_h	1	0.75	0.75	0.75	0.6	0.6	0.6	x	1	1	0
c_i	1	0.75	0.75	0.75	0.6	0.6	0.6	1	x	1	0
c_j	1	0.75	0.75	0.75	0.6	0.6	0.6	1	1	x	0
c_k	0	0	0	0	0	0	0	0	0	0	x

Fig. 9. Initial similarity matrix

In the first iteration there are six pairs of clusters having maximal similarity (similarity = *1*): (c_a, c_h), (c_a, c_i), (c_a, c_j), (c_h, c_i), (c_h, c_j) and (c_i, c_j). Since all the six pairs have the same sum of cluster contents' sizes, any of them can be chosen for merging (the actual choice may depend on a particular implementation of the algorithm). In this example we assume that a pair of clusters first found during a scan of the similarity matrix (performed row after row, from left to right) is chosen in such situations. This leads to selecting (c_a, c_h) as the first pair of clusters to be merged. In the next two iterations clusters having similarity = *1* are merged to form c_{ahij}. The database and the similarity matrix after the third iteration are shown in Figure 10. In the fourth iteration, we merge the clusters c_b and c_e (see Figure 11). In the fifth iteration, the clusters c_{be} and c_c are merged to form the intermediate result presented in Figure 12. Then, the merging of the cluster clusters c_d and c_f in the sixth iteration leads to the state illustrated by Figure 13.

Cluster	Description	Sequences
c_{ahij}	p_1, p_8, p_9, p_{10}	1, 2, 3
c_b	p_2	1, 2, 3, 4
c_c	p_3	1, 2, 3, 6
c_d	p_4	1, 2, 3, 7
c_e	p_5	1, 2, 3, 4, 6
c_f	p_6	1, 2, 3, 4, 7
c_g	p_7	1, 2, 3, 6, 7
c_k	p_{11}	8, 9, 10

	c_{ahij}	c_b	c_c	c_d	c_e	c_f	c_g	c_k
c_{ahij}	x	0.75	0.75	0.75	0.6	0.6	0.6	0
c_b	0.75	x	0.6	0.6	0.8	0.8	0.5	0
c_c	0.75	0.6	x	0.6	0.8	0.5	0.8	0
c_d	0.75	0.6	0.6	x	0.5	0.8	0.8	0
c_e	0.6	0.8	0.8	0.5	x	0.66	0.66	0
c_f	0.6	0.8	0.5	0.8	0.66	x	0.66	0
c_g	0.6	0.5	0.8	0.8	0.66	0.66	x	0
c_k	0	0	0	0	0	0	0	x

Fig. 10. Database and similarity matrix after 3 iterations

Cluster	Description	Sequences
c_{ahij}	p_1, p_8, p_9, p_{10}	1, 2, 3
c_{be}	p_2, p_5	1, 2, 3, 4, 6
c_c	p_3	1, 2, 3, 6
c_d	p_4	1, 2, 3, 7
c_f	p_6	1, 2, 3, 4, 7
c_g	p_7	1, 2, 3, 6, 7
c_k	p_{11}	8, 9, 10

	c_{ahij}	c_{be}	c_c	c_d	c_f	c_g	c_k
c_{ahij}	x	0.6	0.75	0.75	0.6	0.6	0
c_{be}	0.6	x	0.8	0.5	0.66	0.66	0
c_c	0.75	0.8	x	0.6	0.5	0.8	0
c_d	0.75	0.5	0.6	x	0.8	0.8	0
c_f	0.6	0.66	0.5	0.8	x	0.66	0
c_g	0.6	0.66	0.8	0.8	0.66	x	0
c_k	0	0	0	0	0	0	x

Fig. 11. Database and similarity matrix after 4 iterations

Cluster	Description	Sequences
c_{ahij}	p_1, p_8, p_9, p_{10}	1, 2, 3
c_{bce}	p_2, p_3, p_5	1, 2, 3, 4, 6
c_d	p_4	1, 2, 3, 7
c_f	p_6	1, 2, 3, 4, 7
c_g	p_7	1, 2, 3, 6, 7
c_k	p_{11}	8, 9, 10

	c_{ahij}	c_{bce}	c_d	c_f	c_g	c_k
c_{ahij}	x	0.6	0.75	0.6	0.6	0
c_{bce}	0.6	x	0.5	0.66	0.66	0
c_d	0.75	0.5	x	0.8	0.8	0
c_f	0.6	0.66	0.8	x	0.66	0
c_g	0.6	0.66	0.8	0.66	x	0
c_k	0	0	0	0	0	x

Fig. 12. Database and similarity matrix after 5 iterations

Cluster	Description	Sequences
c_{ahij}	p_1, p_8, p_9, p_{10}	1, 2, 3
c_{bce}	p_2, p_3, p_5	1, 2, 3, 4, 6
c_{df}	p_4, p_6	1, 2, 3, 4, 7
c_g	p_7	1, 2, 3, 6, 7
c_k	p_{11}	8, 9, 10

	c_{ahij}	c_{bce}	c_{df}	c_g	c_k
c_{ahij}	x	0.6	0.6	0.6	0
c_{bce}	0.6	x	0.66	0.66	0
c_{df}	0.6	0.66	x	0.66	0
c_g	0.6	0.66	0.66	x	0
c_k	0	0	0	0	x

Fig. 13. Database and similarity matrix after 6 iterations

Cluster	Description	Sequences
c_{ahij}	p_1, p_8, p_9, p_{10}	1, 2, 3
c_{bcdef}	p_2, p_3, p_4, p_5, p_6	1, 2, 3, 4, 6, 7
c_g	p_7	1, 2, 3, 6, 7
c_k	p_{11}	8, 9, 10

	c_{ahij}	c_{bcdef}	c_g	c_k
c_{ahij}	x	0.5	0.6	0
c_{bcdef}	0.5	x	0.83	0
c_g	0.6	0.83	x	0
c_k	0	0	0	x

Fig. 14. Database and similarity matrix after 7 iterations

Cluster	Description	Sequences
c_{ahij}	p_1, p_8, p_9, p_{10}	1, 2, 3
c_{bcdefg}	p_2, p_3, p_4, p_6, p_5, p_7	1, 2, 3, 4, 6, 7
c_k	p_{11}	8, 9, 10

	c_{ahij}	c_{bcdefg}	c_k
c_{ahij}	x	0.5	0
c_{bcdefg}	0.5	x	0
c_k	0	0	x

Fig. 15. Database and similarity matrix after 8 iterations

The intermediate results of the seventh and eighth iterations are presented in Figures 14 and 15. After the eighth iteration, the requested number of clusters is reached and the Merge Phase ends. Then, in the optional Description Pruning Phase descriptions of the discovered clusters are being minimized. The following relations between patterns are true: $p_2 \subset p_1$, $p_3 \subset p_1$, $p_4 \subset p_1$, $p_5 \subset p_1$, $p_6 \subset p_1$, $p_7 \subset p_1$, $p_5 \subset p_2$, $p_6 \subset p_2$, $p_5 \subset p_3$, $p_7 \subset p_3$, $p_6 \subset p_4$, $p_7 \subset p_4$, $p_9 \subset p_8$, and $p_{10} \subset p_8$. This leads to removing p_8 from the description of cluster c_{ahij}, and p_2, p_3, and p_4 from the description of cluster c_{bcdefg} because each of them includes some other pattern from the same description, for example $p_9 \subset p_8$ and they are both in the description of cluster c_{ahij}. After completion of the Description Pruning Phase we get the final result of clustering shown in Figure 16.

Cluster	Description	Customer Sequences
c_{ahij}	p_1 = <(10) (20 30)>, p_9 = <(40)>, p_{10} = <(50)>	1, 2, 3
c_{bcdefg}	p_5 = <(10)>, p_6 = <(20)>, p_7 = <(30)>	1, 2, 3, 4, 6, 7
c_k	p_{11} = <(100)>	8, 9, 10

Fig. 16. Discovered clusters

The algorithm found three clusters, two of which overlap (the content of one of them even includes the content of the other, but their descriptions do not imply that). Data sequence of the customer *5* is not contained in any cluster because it did not support any frequent pattern, which is a consequence of the fact that the customer did not follow any typical buying pattern.

4 Conclusions and Future Work

We considered the problem of clustering sequential data in large databases. Due to the limitations of the existing clustering methods, we introduced the new algorithm, which uses frequent patterns to generate both clustering model and cluster contents. The algorithm iteratively merges smaller, similar clusters into bigger ones until the requested number of clusters is reached. In the absence of a well-defined metric

space, we propose the cooccurrence-based similarity measure to be used in cluster merging. The advantage of the proposed measure is that no additional access to the source database is needed to evaluate the inter-cluster similarity.

In the future, we plan to extend this work along the following lines:

- analysis of the methods for disjoint clusters generation, where each source sequence is allowed to belong to one cluster only,
- classification of sequential data, leading to generation of the classification rules.

References

1. Agrawal R., Srikant R.: Mining Sequential Patterns. Proc. of the 11th Int'l Conference on Data Engineering (ICDE), Taipei, Taiwan (1995)
2. Ester M., Kriegel H-P., Sander J., Xu X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Proc. of the 2nd Int'l Conference on Knowledge Discovery and Data Mining (KDD), Portland, Oregon (1996)
3. Ester M., Kriegel H-P., Xu X.: A Database Interface for Clustering in Large Spatial Databases. Proc. of the 1st Int'l Conference on Knowledge Discovery and Data Mining (KDD), Montreal, Canada (1995)
4. Gibson D., Kleinberg J.M., Raghavan P.: Clustering Categorical Data: An Approach Based on Dynamical Systems. Proc. of the 24th Int'l Conference on Very Large Data Bases (VLDB), New York City, New York (1998)
5. Guha S., Rastogi R., Shim K.: CURE: An Efficient Clustering Algorithm for Large Databases. Proc. of the ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA (1998)
6. Han E., Karypis G., Kumar V., Mobasher B.: Clustering based on association rules hypergraphs. Proc. Workshop on Research Issues on Data Mining and Knowledge Discovery (1997)
7. Han E., Karypis G., Kumar V., Mobasher B.: Hypergraph Based Clustering in High-Dimensional Data Sets: A summary of Results. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Vol.21 No. 1 (1998)
8. Ng R.T., Han J.: Efficient and effective clustering methods for spatial data mining. Proc. of the 20th International Conference on Very Large Data Bases (VLDB) , Santiago de Chile, Chile (1994)
9. Ramkumar G. D., Swami A.: Clustering Data Without Distance Functions. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Vol.21 No. 1 (1998)
10. Srikant R., Agrawal R.: Mining Sequential Patterns: Generalizations and Performance Improvements. Proc. of the 5th Int'l Conference on Extending Database Technology (EDBT), Avignon, France (1996)
11. Zhang T., Ramakrishnan R., Livny M.: Birch: An efficient data clustering method for very large databases. Proc. of the ACM SIGMOD International Conference on Management of Data, Montreal, Canada (1996)