The notion of similarity in data and pattern space†

Irene Ntoutsi[1,2]

[1] Research Academic Computer Technology Institute, 11 Aktaiou & Poulopoulou str, 118 51, Athens, Greece

[2] Department of Informatics, University of Piraeus, 80 Karaoli & Dimitriou str, 185 34, Piraeus, Greece
E-mail: ntoutsi@unipi.gr

Abstract. In the recent years, our ability of collecting information rapidly increases and huge databases that change over time in a high frequency have been developed. On the other hand, the data mining techniques used for extracting essential information from raw data are rather complex, thus not synchronized to underlying data changes. There is a need of detecting whether synchronization operations should take place so as to rescue resources. Motivated from the above problem, we introduce the notion of similarity in data and pattern space and examine their relationship.

1 Introduction

Transactional databases organize collections of transactions where each transaction is described by a specific set of attributes. A representative example is that of a supermarket database, where a transaction lists all items bought by a customer during a single shopping transaction.

An association rule is a probabilistic relationship of the form A→B between sets of database attributes. In the simplest case, the attributes' values are of Boolean type and the database contains a set of records each of which is described by a set of attribute values. Each attribute value declares the presence or absence of each of the attributes in the transaction.

Real market basket databases keep terabytes of information about customers' habits and their data are under continuous changes. Since it is important to keep patterns up to date, the complexity of patterns generation of such a large frequency is extremely high. Obviously, it is critical for these companies (and all of us) to be able to detect whether a regeneration of patterns should take place. In other words, we would like to know whether changes in data space (raw data) would not affect significantly the corresponding pattern space (i.e. patterns already extracted) and when such changes enforce patterns' re-generation.

† Work partially supported by the European Commission under the IST-2001-33058 project PANDA "Patterns for next-generation Database Systems" (2001-04) and the Greek government under the EPEAEK II / Heracletos Program (2003-05).

The paper is organized as follows. In the next section, we discuss the problem of similarity in data and pattern space and associate it with the assignment problem of the graph theory domain. In Section 3, we refer to some concepts of graph theory domain that are used during this work. In Section 4, we analyze the notion of similarity in data and pattern space. Section 5 presents experimental results, whereas section 6 concludes, giving also hints for future work.

2 Problem Definition

Consider two transactional datasets D_1 and D_2 of equal cardinality[2], i.e., $|D_1|=|D_2| = n$. Each dataset is described by a specific ordered set of attributes $(a_1, a_2, ..., a_k)$ and a Boolean value (0/1) is assigned to each attribute. Every pair (i, j), where $i \in D_1$ and $j \in D_2$, is associated with a weight w, $0 \leq w \leq k$, that denotes the *distance* between elements i and j. Strictly speaking, the difference between two items $i \in D_1$ and $j \in D_2$ equals the number of permutations required so as to jump from i to j. For example, *distance* (110, 010) = 1.

We wish to find the similarity between the two datasets, in terms of *optimal matching* between elements of D_1 and D_2, so as every element of D_1 and D_2 participates at one and only one matching pair. With the term 'optimal' we mean a matching that minimizes the aggregate difference between the two datasets (i.e. the sum of differences between pairs of items from D_1 and D_2).

The above analysis also holds for the pattern space (during this work we use association rules as examples of patterns). Consider two pattern sets P_1 and P_2 of equal cardinality[3], i.e., $|P_1|=|P_2| = m$. Each pattern set is described by a specific ordered set of items $(i_1, i_2, ..., i_l)$ and within each pattern P every item i is assigned a value (0/ H/ B); H, if i appears at the LHS of P; B, if i appears at the RHS of P; 0, otherwise. Every pair (i, j), where $i \in P_1$ and $j \in P_2$, is associated with a weight w, $0 \leq w \leq l$, that denotes the *distance* between patterns i and j. The distance is computed as the aggregated distance between the corresponding items of patterns i and j. For example, $distance(\{a_1 \rightarrow a_2\},\{a_1 \rightarrow a_2\}) = 0$

Next we present two examples (in data and pattern space) that demonstrate the representation we adopt.

Example 1

Consider two datasets $D_1=\{\{a_1, a_2\}; \{a_2, a_3\}; \{a_2\}\}$ and $D_2=\{\{a_1, a_2, a_3\}; \{a_3\}; \{a_1, a_2\}\}$. Each dataset consists of $n = 3$ transactions. The total number of attributes equals 3, i.e. $k = 3$. The datasets are represented as below:

[2] The above approach could be extended to include the case of datasets of different size as well, by simply adding empty transactions (with zero items each) to the smallest dataset.

[3] As before, the approach could be extended to include the case of pattern sets of different size as well.

Items / Baskets	a_1	a_2	a_3
$\{a_1,a_2\}$	1	1	0
$\{a_2,a_3\}$	0	1	1
$\{a_2\}$	0	1	0

(a)

Items / Baskets	a_1	a_2	a_3
$\{a_1,a_2,a_3\}$	1	1	1
$\{a_3\}$	0	0	1
$\{a_1,a_2\}$	1	1	0

(b)

Fig. 1. Example 1 – representation of transactional data **(a)** D_1 **(b)** D_2

Example 2

Consider two pattern sets $P_1=\{(i_1 \rightarrow i_2),(i_2 \rightarrow i_3)\}$ and $P_2=\{(i_1 \rightarrow i_3),(i_2 \rightarrow i_1,i_3)\}$ derived from the above datasets. Each pattern set consists of $m = 2$ elements. The total number of items equals 3, i.e. $l = 3$. The two pattern sets are represented as below:

Items / Patterns	a_1	a_2	a_3
$\{i_1\} \rightarrow \{i_2\}$	H	B	0
$\{i_2\} \rightarrow \{i_3\}$	0	H	B

(a)

Items / Patterns	a_1	a_2	a_3
$\{i_1\} \rightarrow \{i_3\}$	H	0	B
$\{i_2\} \rightarrow \{i_1,i_3\}$	B	H	B

(b)

Fig. 2. Example 2 - representation of association rule patterns **(a)** P_1 **(b)** P_2

We use the notion of similarity in data and pattern space in order to discover how changes in data space affect the pattern space. The question that we aim to address is twofold: (a) Is there any correlation between the evolution in data space and that in pattern space? (b) If yes, is there any bound where changes in data space do not result to analogous changes in the pattern space derived from these data? Supposed we determined such a behavior, we could make an efficient synchronization of running data mining techniques depending on the evolution of the underlying database.

The problem of finding the similarity between two datasets D_1 and D_2 (respectively, P_1 and P_2 in the pattern space), looks similar to one well-known problem of the graph theory domain, the so-called *assignment problem*. We present this problem and relevant solutions, in the next section.

3 Graph Theory Domain – The Assignment Problem

In this section, we give some definitions concerning the graph theory concepts that are used later in the paper.

A *bipartite graph* is a graph $G(V,E)$ whose vertex set *V* can be split in two non-empty, disjoint sets, *A* and *B* in such a way that every edge of *G* joins a vertex of *A* to a vertex of *B*.

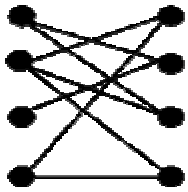


Fig. 3. Example of a bipartite graph

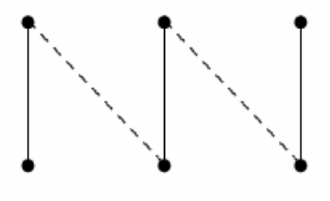


Fig. 4. Example of a perfect matching

A subset *M* of *E* is called a *matching* in *G* if no vertex is incident to more than one edge in *M*. If every vertex of *G* is incident to some edge of *M*, then *M* is called a *perfect matching*.

A graph *G* is *weighted* if we give a cost function *c* that associates each edge with a real value, that is, *c: E → R*. Let *X* be a subset of *E*. The cost of set *X* is:

$$c(X) = \sum_{a \in X} c(a) \quad \textbf{(1)}$$

The problem of similarity could be reduced to a well-known problem of the graph theory domain; the *assignment problem* of finding a *perfect minimum cost weighted matching in bipartite graphs.*

A solution to this problem has been developed by James Munkres and is of $O(n^3)$ runtime complexity [1]. The algorithm, known as *Hungarian method*, takes as input a matrix of the weights of the edges that relate the two disjoint sets of the bipartite graph (we call it distance matrix) and outputs a cover of minimum cost.

Except for the Hungarian method, the graph theory domain has to demonstrate a large number of algorithms for the solution of the assignment problem. Below we present a table of these algorithms, categorized according to their type (sequential or parallel).

Sequential algorithms		
Date	**Authors**	**Complexity**
1955	Kuhn [1]	$O(mn^2)$
1972	Edmonds and Karp [6]	$O(mn \log n)$
1984	Fredman and Tarjan [6,7]	$O(mn + n^2 \log n)$
1985	Gabow [8]	$O(mn^{3/4} \log U)$
1989	Gabow and Tarjan [9]	$O(mn^{1/2} \log nU)$

Date	Authors	Complexity	Processors
1988	Goldberg, Plotkin and Vaidya [10]	$O(n^{2/3}\log^3 n\log nU)$	$O(n^3/\log n)$
1988	Gabow and Tarjan [11]	$O((mn^{1/2}/p)\log p\log nU)$	$O(m/(n^{1/2}\log^2 n))$

Fig. 5. Perfect minimum cost weighted matching problem in bipartite graphs (n: number of vertices; m: number of edges, U: the greatest absolute value among edge costs, p: the number of processors). [3]

According to the first table, the best sequential algorithm is the Scaling and Approximation Algorithm of Gabow and Tarjan with complexity: $O(mn^{1/2}\log nU) \rightarrow O(n\log nU)$ in our case due to $m = n^2$. According to the second table, the best parallel algorithm is the Scaling and Approximation Parallel Algorithm of Gabow and Tarjan with complexity: $O((mn^{1/2}/p)\log p\log nU) \rightarrow O((n/p)\log p\log nU)$ in our case due to $m = n^2$.

4. Similarity

The notion of similarity in data (pattern) space is quite important since it expresses whether two data (pattern) sets share common characteristics. This indication is critical in a variety of applications. Consider, for example, the synchronization between two datasets. The tasks involved in the synchronization process are of large complexity and should be performed only if the corresponding datasets differ significantly. A similar example could be used for the pattern space.

Apart from its discrete value in data and pattern space, the similarity is also important in activities that involve both data and pattern space. Consider the example of a company that generates a patterns' set p from a specific dataset d. It's important for the company to keep patterns up to date. Since data change over time the company has to re-extract patterns even if changes in data are inconsiderable. This operation is of large complexity and should be used wisely in order to rescue resources. From the above, what emerges is the need for determining a limit that would demonstrate whether the regeneration of patterns should take place.

In the next sections we refer to the notion of similarity in data and pattern space. As we have already mentioned, we reduce this problem to the assignment problem of the graph theory domain and use the Hungarian method in order to solve it.

4.1 Similarity in data space

We have already referred to the representation of datasets. The dissimilarity between two datasets D_1 and D_2 is represented through a distance matrix. Each element (i,j) of the distance matrix represents the difference between items i and j (i.e. the number of

permutations required to jump from i to j). More formally, distance(i, j) = XOR(i, j), $i \in D_1$ *and* $j \in D_2$. The total dissimilarity count between the two datasets equals to the sum of dissimilarity counts between the items that comprise the matching pairs.

Let's consider the datasets of Example 1. Their dissimilarity matrix is depicted below:

	111	**001**	**110**
110	1	3	0
011	1	1	2
010	2	2	1

Fig. 6. Example 1 - the distance matrix (data space)

The above matrix constitutes the input of the Hungarian method used for finding the optimal minimum cost matching between D_1 and D_2. Its output is depicted underlined in the above figure and so *distance*(D_1, D_2) = 1+1+1=3.

4.2 Similarity in pattern space

We follow a strategy similar to that used in data space. The distance matrix contains the dissimilarity between the two pattern sets P_1 and P_2. Each element *(i, j)* of the distance matrix represents the difference between patterns i and j. According to values 0/ H/ B that are assigned to each item of the patterns, the difference is as follows:

		Possible values for i ∈ P1		
		0	**LHS**	**RHS**
Possible values for $j \in P_2$	**0**	0	1	1
	LHS	1	0	k
	RHS	1	k	0
	RHS	1	k	0

Fig. 7. Computing similarity in pattern space

Variable k denotes the dissimilarity between two association rules that share some common item, i.e. when an item of the LHS of the first rule participates in the RHS of the second rule and vice versa. Intuitively, two association rules that share some item in their LHS or RHS are more similar than two association rules that "alternate" the same item between the LHS of the first and the RHS of the second or vice versa. Assume, for example, three association rules: AR_1 = *{milk}→{bread}*, AR_2 = *{milk}→{beer}*, and AR_3 = *{beer}→{milk}*. According to the above table, *dissimilarity*(AR_1, AR_2) < *dissimilarity*(AR_1, AR_3). Since field experts define the dissimilarity semantics, we represent it as a parameter k *(*$0 \leq k \leq 1$*)* to be tuned.

Let us now consider the pattern sets of the Example 2 (k = 0.5). Their dissimilarity matrix is depicted below:

	HBO	**0HB**
HOB	2	2
BHB	3	1

Fig. 8. Example 2 - the distance matrix (pattern space), $k = 0.5$

The above matrix constitutes the input of the Hungarian method. Its output is depicted underlined in the above figure and so $distance(P_1, P_2) = 2 + 1 = 3$.

5. Experimental Results

In order to evaluate the applicability of our approach we generated synthetic transactional data. The generator takes as input the number of items per transaction and the number of transactions we wish to generate and randomly decides whether an item is included (value 1) or not (value 0) in a transaction. In our experiments we use datasets of 100 transactions where each transaction consists of 10 items.

For the generation of pattern sets we use the well-known Apriori algorithm **[4]** with parameters: *support* = 20% and *confidence = 70%.*

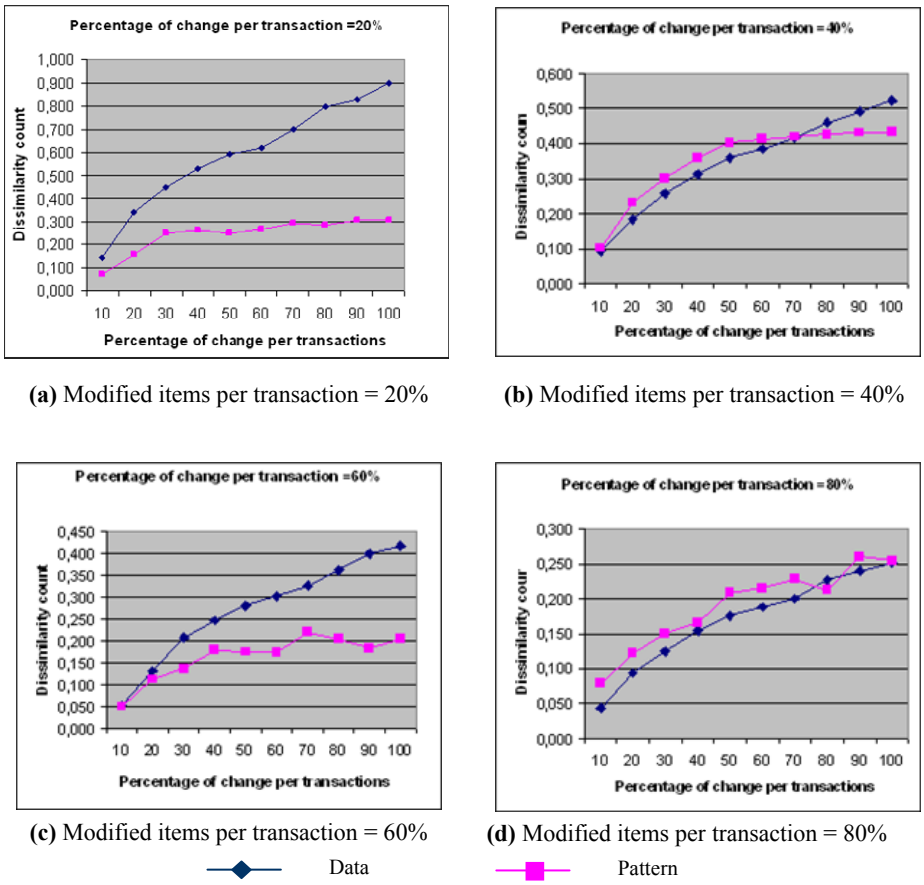
Referring to *k*, the parameter of dissimilarity in pattern space, we fixed it at 0.5 since we saw that its value doesn't affect the behavior of the dissimilarity. However, we do not present these experiments in the paper due to space limitations.

Running of an experiment includes the following steps (in pseudo code form):

```
Generate a synthetic transaction data set D
Extract patterns from D → P
Step = 10
While step ≤ 100
{
          change step% transactions of D → D'
          extract patterns from D' → P'
          compute dissimilarity(D, D') //in data space
computer dissimilarity(P, P') //in pattern space
          step += 10
}
```

Apart from experimenting with the percentage of change per transactions (i.e. the number of modified transactions), we also experimented with the percentage of change per transaction (i.e. the number of modified items per transaction).

For more objective results, we run each experiment 10 times and used the average values. The experimental results are presented below:



(a) Modified items per transaction = 20% **(b)** Modified items per transaction = 40%

(c) Modified items per transaction = 60% **(d)** Modified items per transaction = 80%

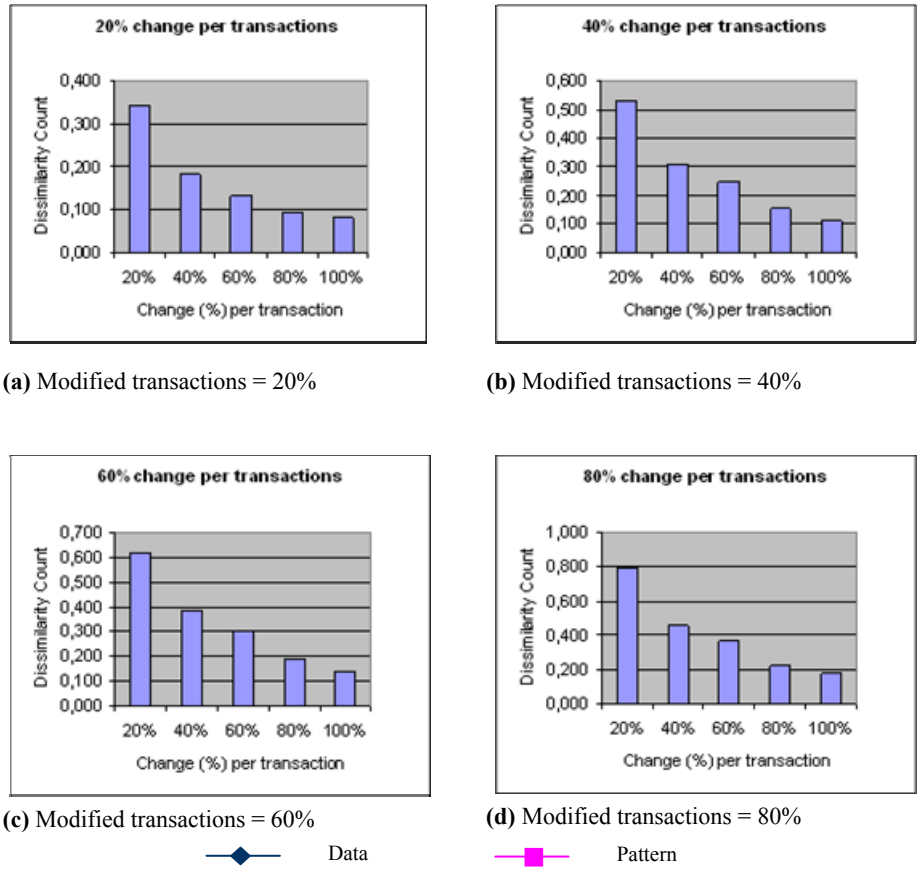
Fig. 9. The behavior of dissimilarity with respect to the number of modified transactions

Figure 9 shows the dissimilarity' behavior as the number of modified transactions increases. At each experiment we changed 20%, 40%, 60%, 80% of the transactions and fixed the number of modified items per transaction. For example, in Figure 9 (a) we alternated 20% of the items of each of the modified transaction (2 items in our case).

It is apparent from Figure 9 that the dissimilarity in data space grows linearly as the percentage of change per transactions increases. So, starting from a dataset D and changing 20% (D_20) , 40% (D_40), …, 80% (D_80) of its transactions, the result of the comparison between (D, D_20), (D, D_40), … , (D, D_80) increases. This result was expected, as the more we change a dataset the more it would differ from its initial state. The maximum dissimilarity in data space is given by: $n*p_1*k*p_2$, where n: number of transactions, p_1: percentage of change per transactions, m: number of attributes per transaction and p_2: percentage of change per transaction. For example,

in Figure 9 (a) the minimum dissimilarity count is 20 (for p_1 = 10%) and the maximum distance is 200 (for p_1 = 100%).

The behavior in pattern space is quite different. As it seems from Figure 9, the dissimilarity in pattern space increases linearly at initial steps and is afterwards stabilized. The critical limit seems to be above the 50%; i.e. when the new dataset is 50% different from the initial dataset it seems that dissimilarity count is stabilized. The maximum dissimilarity count in pattern space is given by: *m*l*, where *m*: maximum number of patterns and *l*: maximum number of items per pattern.



(a) Modified transactions = 20%

(b) Modified transactions = 40%

(c) Modified transactions = 60%

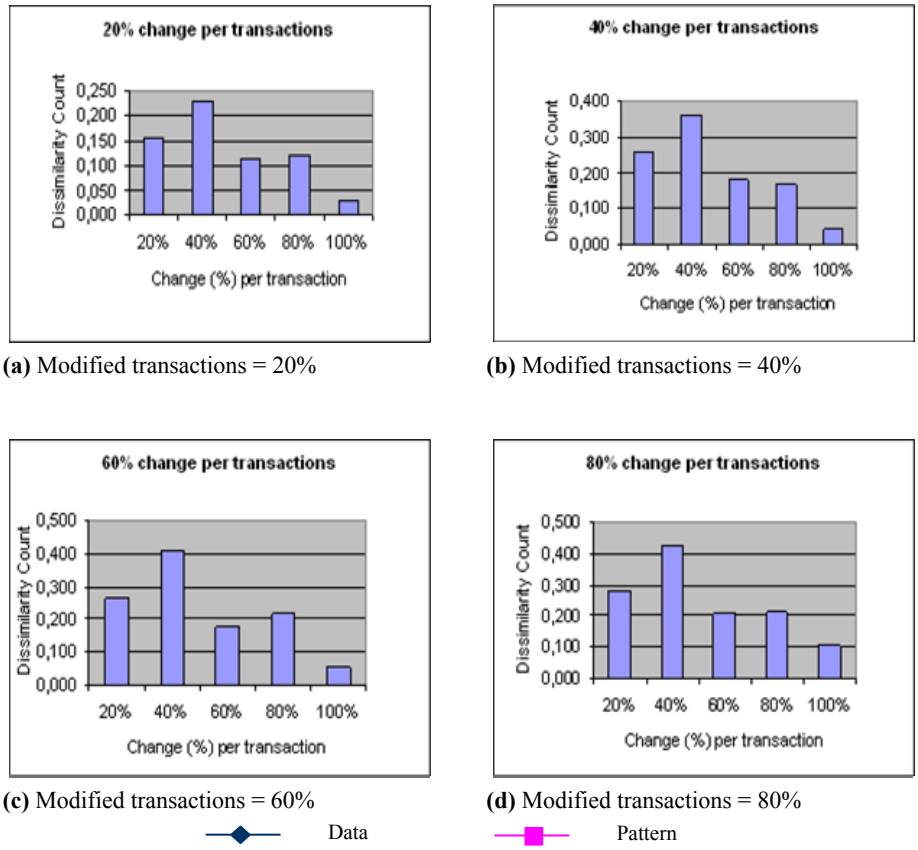
(d) Modified transactions = 80%

Fig. 10. The behavior of dissimilarity with respect to the number of modified items per transaction (data space)

Figure 10 illustrates dissimilarity's behaviour as the number of modified items per transaction increases. At each experiment we changed 20%, 40%, …, 100% of items per transaction and fixed the number of modified transactions. For example, in Figure 10 (a) we modified 20% of transactions (20 transactions in our case).

As illustrated in Figure 10, the behavior of dissimilarity is reproduced independently of the number of modified transactions. The maximum dissimilarity

(i.e. minimum similarity) is achieved at initial levels where the percentage of change per transaction is small, whereas the minimum dissimilarity (i.e. maximum similarity) is achieved at the last step where each of the modified transactions changes completely. This result was not what we expected; we thought that the more we change some transactions the more the dissimilarity will grow. However, it seems that the Hungarian method we used for the computation of dissimilarity rearranges the matches so as to minimize the overall dissimilarity count.



(a) Modified transactions = 20%

(b) Modified transactions = 40%

(c) Modified transactions = 60%

(d) Modified transactions = 80%

Fig. 11. The behavior of dissimilarity with respect to the number of modified items per transaction (pattern space)

Figure 11 illustrates how dissimilarity's behavior in pattern space is affected as the number of modified items per transaction increases. At each experiment we changed 20%, 40%, …, 100% of items per transaction and fixed the number of modified transactions. For example, in Figure 11 (a) we modified 20% of transactions (20 transactions in our case).

Again, the behavior is reproduced (the same happens in data space). The maximum dissimilarity is achieved where the change per transaction equals to 40% and the smallest where the change per transaction equals to 100%.

6. Conclusions and Future work

In this paper we discussed the notion of similarity in data and pattern space. We reduced the problem of similarity to one well-known problem of the graph theory domain, the assignment problem and used the Hungarian method in order to find an optimal matching of minimum cost.

The experimental results showed that although dissimilarity in data space grows linearly as the percentage of modified transactions increases, dissimilarity in pattern space stabilizes after initial steps (experiments showed than this limit is near 50% of changes in initial data). This conclusion is important since it denotes that there is an upper bound in data space changes above which the corresponding patterns do not differ significantly from already extracted patterns.

The work presented in this paper is at preliminary steps. Our future work, would involve performance issues. As we saw, the complexity of finding a perfect weighted matching between two datasets using the Hungarian method is high ($O(n^3)$). There are however better solutions, like the parallel algorithm introduced by Gabow and Tarjan with complexity $O((n/p)\log p \log nU)$. We plan to explore more efficient methods for the computation of dissimilarity. Except for the complexity of computing the dissimilarity, we have to take into account the space complexity of the solution; the Hungarian method requires n^2 space for the dissimilarity matrix. Unfortunately, the above mentioned costs increase proportionally with the size of data (pattern) space.

Our work should be extended in order to take into account concept hierarchies that might exist between items. For example, the dissimilarity between two rules that differ on a single item that is the particular brand of beer (say, Heineken and Amstel) is intuitively less than the dissimilarity between two rules that differ on a single item that is totally separate (say, Heineken and Pampers) [5].

As a second direction of work, although currently we compute the similarity between two patterns by taking into account only their structural components (LHS and RHS), it is natural that two identical (referring to structure) association rules that differ on the measure values (confidence and support) cannot be considered as identical (i.e., having *dissimilarity* = 0). We should explore the possibilities of incorporating the measure component in the computation of similarity [5].

References

1. H.W.Kuhn, *The Hungarian method for the assignment problem*, Naval Res. Logist. Quart. 2: 83-97, 1955
2. The PANDA Project, http://dke.cti.gr/panda, 2002
3. H.A. Baier Saip and C.L. Lucchesi, Matching algorithms for bipartite graph. Technical Report DCC-03/93, Departamento de Cincia da Computao, Universidade Estudal de Campinas, 1993.
4. C. Borgelt, Apriori Implementation, http://fuzzy.cs.uni-magdeburg.de/~borgelt/
5. Y. Theodoridis, Proceedings of PANDA Workshop on Pattern Base Management Systems, PANDA Technical Report T.R.-2003-02, Como, Italy.

6. J. Edmonds and R.M.Karp. Theoritical Improvements in algorithmic efficiency for network flow problems. Journal of the Assoc. for Comput. Mach., 19(2): 248-264, 1972
7. M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in imporved network optimization algorithms.In 25th FOCS, pages 338-346 1984.
8. H. N. Gabbow. Scaling algorithms for network problems, Journal of Comput. and Syst. Sci., 31(2): 148-168, Oct. 1985.
9. H. N. Gabbow and R. E. Tarjan. Faster scaling algorithms for network problems, SIAM J. Comput., 18(5): 1013-1036, Oct. 1989.
10. A. V. Goldberg, S. A. Plotkin, and P. M. Vaidya. Sublinear-time parallel algorithms for matching and related problems. In 29th FOCS, pages 174-185, 1988
11. H. N. Gabow and R. E. Tarjan. Almost-optimal speed-ups of algorithms for matchoing and related problems. In 20th STOC, pages 514-527, 1988.