Data & Knowledge Engineering 68 (2009) 244–260

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

# The PANDA framework for comparing patterns ☆

Ilaria Bartolini [a], Paolo Ciaccia [a], Irene Ntoutsi [b], Marco Patella [a,*], Yannis Theodoridis [b]

[a] DEIS, University of Bologna, viale Risorgimento, 2, 40136 Bologna, Italy
[b] Department of Informatics, University of Piraeus, Greece and Research Academic Computer Technology Institute, Athens, Greece

ARTICLE INFO

*Article history:*
Received 10 July 2008
Received in revised form 3 October 2008
Accepted 4 October 2008
Available online 25 October 2008

*Keywords:*
Pattern comparison
Pattern base management systems
Data models
Knowledge discovery

ABSTRACT

Data Mining techniques are commonly used to extract *patterns*, like association rules and decision trees, from huge volumes of data. The comparison of patterns is a fundamental issue, which can be exploited, among others, to synthetically measure dissimilarities in evolving or different datasets and to compare the output produced by different data mining algorithms on a same dataset. In this paper, we present the PANDA framework for computing the dissimilarity of both *simple* and *complex* patterns, defined upon raw data and other patterns, respectively. In PANDA the problem of comparing complex patterns is decomposed into simpler sub-problems on the component (simple or complex) patterns and so-obtained partial solutions are then smartly aggregated into an overall dissimilarity score. This intrinsically recursive approach grants PANDA with a high flexibility and allows it to easily handle patterns with highly complex structures. PANDA is built upon a few basic concepts so as to be generic and clear to the end user. We demonstrate the generality and flexibility of PANDA by showing how it can be easily applied to a variety of pattern types, including sets of itemsets and clusterings.

© 2008 Elsevier B.V. All rights reserved.

## 1. Introduction

A huge amount of heterogeneous data is collected nowadays from a variety of data sources (e.g., business, health care, telecommunication, science). The storage rate of these data collections is growing at a phenomenal rate (over 1 exabyte per year, according to a recent survey [1]). Due to their quantity and complexity it is impossible for humans to thoroughly investigate these data collections through a manual process. Knowledge discovery in data (KDD) tries to solve this problem by discovering hidden information using data mining (DM) techniques. DM results, called *patterns*, constitute compact and rich in semantics representations of raw data [2]. Well-known examples of patterns are decision trees, clusterings, and frequent itemsets. Patterns reduce the complexity and size of data collections, while preserving most of the information of the original raw data; the degree of preservation, however, strongly depends on the parameters of the DM algorithms used for their extraction.

The wide spreading of DM technology makes the problem of efficiently managing patterns an important research issue. Ideally, patterns should be treated by pattern management systems as "first-class citizens", in the same fashion that raw data are treated by traditional database management systems. Along this line of research some interesting results, mainly concentrated on representation and querying issues, have been obtained [2,3]. In this paper, we address the relevant issue of *pattern comparison*, i.e., how to establish whether two patterns are similar or not. Pattern comparison is valuable in

☆ This work was partially supported by the IST-2001-33058 Thematic Network PANDA "PAtterns for Next-generation DAtabase systems".
* Corresponding author.
*E-mail addresses:* i.bartolini@unibo.it (I. Bartolini), paolo.ciaccia@unibo.it (P. Ciaccia), ntoutsi@unipi.gr (I. Ntoutsi), marco.patella@unibo.it (M. Patella), ytheod@unipi.gr (Y. Theodoridis).

0169-023X/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2008.10.004

monitoring and detecting changes in patterns describing evolving data (e.g., the purchasing behavior of customers over time), as well as in a number of other scenarios, some of which are sketched in Section 1.1.

A principled approach to pattern comparison needs to address several problems. First, there is a large amount of heterogeneous patterns for which a dissimilarity operator should be defined: since each of these pattern types could have its own specific requirements on how the dissimilarity should be assessed, it seems almost impossible (and possibly meaningless) to define a "universal" dissimilarity measure. Second, besides patterns defined over raw data (hereafter called *simple patterns*), there also exist patterns defined upon other patterns, e.g., a cluster of frequent itemsets, an association rule of clusters, a forest of decision trees, etc. For these patterns, hereafter called *complex patterns*, dissimilarity operators should also be defined: how these are related to the corresponding ones defined for component patterns needs to be addressed. Third, one should consider that two patterns can be more or less similar both in the data they represent and in the way they represent such data. For instance, two clusters might differ either because of their "shape" or because of the amount of raw data they summarize (or because of both).

Given the above, we can state a series of high-level methodological requirements that a framework for dissimilarity assessment should satisfy:

*General applicability:* The framework should be applicable to arbitrary types of patterns.
*Flexibility:* The framework should allow for the definition of alternative dissimilarity functions, even for the same pattern type. Indeed, the end user should be able to easily adjust the dissimilarity criterion to her specific needs.
*Simplicity:* The framework should be built upon a few basic concepts, so as to be understandable to the end user.
*Efficiency:* It should be possible to define the dissimilarity between patterns without the need of accessing the underlying raw data. This requirement also encompasses privacy issues, e.g., when raw data are not publicly available.

The framework we propose, called PANDA,[1] addresses above requirements as follows. Generality is achieved by considering that patterns can be (recursively) defined by means of a set of type constructors. To gain the necessary flexibility in defining dissimilarity operators, PANDA adopts a modular approach. In particular, the problem of comparing complex patterns is reduced to the one of comparing the corresponding sets (or lists, etc.) of component (simpler) patterns. Component patterns are first paired (using a specific *matching type*) and their scores are then aggregated (through some *aggregation function*) so as to obtain the overall dissimilarity score. This recursive definition of dissimilarity allows highly complex patterns to be easily handled and, due to modularity, to change any component with an alternative one. To address the requirement of simplicity, PANDA adopts a consistent approach to model patterns, which are viewed as entities composed of two parts: the *structure* component identifies "interesting" regions in the attribute space, e.g., the head and the body of an association rule, whereas the *measure* component describes how the pattern is related to the underlying raw data, e.g., the support and the confidence of the rule. When comparing two simple patterns, the dissimilarity of their structure components (hereafter, *structure dissimilarity*) and the dissimilarity of their measure components (hereafter, *measure dissimilarity*) are combined (through some *combining function*) in order to derive the total dissimilarity score. Finally, considering the efficiency issue, PANDA only works in "pattern space", i.e., raw data need not to be accessed to evaluate patterns' dissimilarity.

It has to be remarked that is *not* in the PANDA scope the issue of determining the "best" measure for every comparison problem. Indeed, PANDA represents a *conceptual* environment within which specific, user- and/or application-dependent, dissimilarity measures can be framed. Obviously, PANDA is also amenable to act as a *software* framework, in which case further advantages are that of favoring the reusability of components and the easy development of user-defined building blocks into ready-to-use libraries.

### 1.1. Motivating examples

In this section, we provide some illustrative examples which demonstrate the usefulness of a pattern comparison operation. A first application is as an alternative to the comparison of raw data collections, e.g., the monthly sales of a supermarket. Approaches which use pattern sets in order to compare the original raw datasets already exist in the literature, e.g. [4,5]: such approaches are based on the intuition that, since patterns condensate the information existing in the raw data, their dissimilarity is a (either lossless or lossy) representation of the dissimilarity of the originating data [6]. Defining such a mapping between dissimilarity in the raw data space and that in the corresponding pattern space is really useful: if the comparison between patterns does not show substantial differences, it is possible to avoid a thorough (and costly) analysis on the raw datasets. In the same direction, pattern comparison might be helpful in the distributed database domain to analyze, for example, differences of data characteristics across distributed datasets (e.g., customer transactions in branches of a supermarket or human reactions to chemical/biological substances). Other applications include pattern base synchronization (i.e., keeping patterns up to date with respect to the original raw data), versioning support in a pattern management system (getting a differential backup of the new version or compare versions of the pattern base so as to discover changes and outliers), the discovery of unexpected or outlier patterns (by comparing them to a target pattern), the evaluation of DM

[1] PANDA stands for PAtterns for Next-generation DAtabase systems, an acronym used for the IST-2001-33058 project of the European Union, which proposed and studied the PBMS (Pattern Base Management Systems) concept.

algorithms (through the comparison of their outcomes), or secure DM where, due to privacy considerations, only patterns (and not the underlying raw data) are available; in this latter case, the comparison should involve only pattern space characteristics, since connection to raw data is lost.

We conclude this section by describing a few scenarios where similarity between patterns plays an important role.

**Example 1.** Consider a telecommunication company providing a package of new generation services with respect to different customer profiles. Let a decision maker of the company request a monthly report depicting the aggregated usage information of this package as extracted from a data warehouse. Such a report would be far more translatable by the decision maker, e.g., for target marketing, if it was accompanied by the monthly comparison of the classification of the customer profiles using such services, as these are portrayed, say, via decision tree models.

**Example 2.** A spatial DM application analyzes how much the density of population in a town correlates with the number of car accidents. For privacy reasons, raw data are not available, rather only the distributions of population and car accidents in the areas of the town can be used. Such distributions cannot only be compared on a per-area basis, because a high correlation is only detected when the distributions of neighboring areas are compared. The definition of a similarity operator between distributions should be flexible enough to take such correlation into account.

**Example 3.** A copy detection system developer has to experiment with different techniques for comparing multimedia documents, in order to select the most effective one. She is given a feature-based representation of the documents (e.g., list of keywords with weights for the text, distribution of color for the images), and needs to setup a set of methods that take into account all such features and return a score assessing how similar two documents are.

The rest of the paper is organized as follows: in Section 2, we describe the pattern model underlying the framework and introduce two running examples. Section 3 is devoted to explain the basic concepts and mechanisms of the PANDA framework, whereas Section 4 demonstrates how several comparison measures proposed in the literature can be modeled within the framework. Further examples are included in Appendix A, together with actual experimental results as obtained from a prototype software implementation described in Section 5. Related work is discussed in Section 6, while Section 7 concludes.[2]

## 2. Pattern representation

Our approach to pattern representation builds upon the logical pattern base model proposed in [9]; in the sequel we describe only the parts of the model relevant to our purposes (for a detailed presentation, please refer to [9]).

The model assumes a set of *base types* (e.g., `Int`, `Real`, `Boolean`, and `String`) and a set of *type constructors*, including list ($< \cdots >$), set ($\{\cdots\}$), array ($[\cdots]$), and tuple ($(\cdots)$). Let us call $\mathscr{T}$ the *set of types* including all the base types and all the types that can be derived from them through repeated application of the type constructors. Types to which a (unique) name is assigned are called *named types*. Some examples of types are:

| | |
|---|---|
| `{Int}` | set of integers |
| `XYPair = (x:Int,y:Int)` | named tuple type with attributes `x` and `y` |
| `<XYPair>` | list of `XYPair`s |

**Definition 1** (Pattern type). A pattern type is a named pair, $PT = (SS, MS)$, where $SS$ is the *structure schema* and $MS$ is the *measure schema*. Both $SS$ and $MS$ are types in $\mathscr{T}$. A pattern type $PT$ is called *complex* if its structure schema $SS$ includes another pattern type, otherwise $PT$ is called *simple*.

The structure schema $SS$ defines the *pattern space* by describing the structure of the patterns which are instances of the particular pattern type. The complexity of the pattern space depends on the expressiveness of the typing system $\mathscr{T}$. The measure schema $MS$ describes measures that relate the pattern to the underlying raw data or, more in general, provides quantitative information about the pattern itself. It is clear that the measure complexity also depends exclusively on $\mathscr{T}$.

A pattern is an instance of a pattern type, thus it instantiates both the structure and the measure schemas. Assuming that each base type $B$ is associated with a set of values $dom(B)$, it is immediate to define values for any type in $\mathscr{T}$.

**Definition 2** (Pattern). Let $PT = (SS, MS)$ be a pattern type. A *pattern* $p$, instance of $PT$, is defined as $p = (s, m)$, where $p$ is the pattern identifier, $s$ (the *structure* of $p$, also denoted as $p.s$) is a value for type $SS$, $p.s \in dom(SS)$, and $m$ (the *measure* of $p$, also denoted as $p.m$) is a value for type $MS$, $p.m \in dom(MS)$.

Before describing the main concepts of the PANDA framework, we introduce here two running examples that will be used throughout the paper to show the applicability of our framework to real cases, namely the comparison of clusterings and of collections of documents. In particular, in the first example, each clustering (set of clusters) represent an image in a

[2] This paper extends the concepts introduced in [7,8] by providing a more formal presentation, along with a significant number of new experiments and examples of application of the framework.

region-based image retrieval (RBIR) system, where images are retrieved according to their similarity to a provided query image. Experiments on both running examples are detailed in Appendix A.

**Example 4** (Clusterings (images)). We illustrate here the case of the WINDSURF image retrieval system [10], which applies a clustering algorithm on visual characteristics of images so as to divide each image into regions of homogeneous pixels (clusters), but the behavior of other RBIR systems can be modeled in a similar way. In details, WINDSURF applies a Discrete Wavelet Transform to each image and the *k*-means algorithm is used to cluster together pixels sharing common visual characteristic, like color and texture. Each region is then represented as a cluster using the centroid and the corresponding covariance matrix for each color channel and wavelet sub-band (details can be found in [10]), while the cluster support (i.e., the fraction of image pixels contained in the region) is used as the pattern measure.

In terms of the PANDA model, each region (simple pattern) is modeled as

$$\texttt{Region} = (SS : (\texttt{bands} : [(\texttt{center} : [\texttt{Real}]_1^3, \texttt{cov} : [[\texttt{Real}]_1^3]_1^3)]_1^4), MS : (\texttt{supp} : \texttt{Real})).$$

Images are then defined as sets of regions (clusters) with no measure:

$$\texttt{Image} = (SS : \{\texttt{Region}\}, MS : \perp),$$

where $\perp$ denotes the *null type*.

**Example 5** (Collections of documents). The problem of comparing collections of documents is quite common in web mining where, for example, it is used to find sites selling similar products.

The problem, in its basic form, assumes a collection (set) of textual documents, where each document consists of a set of keywords. Each keyword *k* in a document is associated to its (normalized) weight in the document itself (e.g., representing its frequency using *tf*/*idf* measures), and can therefore be modeled as a simple pattern:

$$\texttt{Keyword} = (SS : (\texttt{term} : \texttt{String}), MS : (\texttt{weight} : \texttt{Real})).$$

A possible instance of this type is

$$p407 = ((\texttt{term} = \textit{database}), (\texttt{weight} = 0.5)).$$

Consequently, documents and collections are represented respectively as

$$\texttt{Document} = (SS : \{\texttt{Keyword}\}, MS : \perp),$$
$$\texttt{Collection} = (SS : \{\texttt{Document}\}, MS : \perp).$$

## 3. The PANDA framework

In this section, we provide a framework for assessing the dissimilarity of two patterns, $p_1$ and $p_2$, of the same type *PT*. From Section 2, it is evident that the complexity of *PT* can widely vary and is only restricted by the adopted typing system $\mathscr{T}$.

Our framework is built upon two basic principles:

1. The dissimilarity between two patterns should be evaluated by taking into account both the dissimilarity of their structures and the dissimilarity of their measures.
2. The dissimilarity between two complex patterns should (recursively) depend on the dissimilarity of their component patterns.

The first principle is a direct consequence of having allowed for arbitrarily complex structures in patterns. Since the structure of a complex pattern might include measures of its component patterns, neglecting the structure dissimilarity could easily result in misleading results. For instance, comparing two `Image`s, as defined in Example 4, obviously needs to take into account the structure component, since the measure one is empty. Another motivation underlying this principle arises from the need of building an *efficient* framework, which does not force accessing the underlying dataset(s) in order to determine the dissimilarity of two patterns, e.g., in terms of their common instances. To this end, we use all pieces of information that are available in the pattern space, namely the structural description of the patterns and their quantitative measures with respect to the underlying raw data.

The second principle provides the necessary flexibility to the PANDA framework. Although, for the case of complex patterns, one could devise arbitrary models for their comparison, it is useful and, at the same time, sufficient for practical purposes, to consider solutions that decompose the "difficult" problem of comparing complex patterns into simpler subproblems like those of comparing simple patterns, and then "smartly" aggregate the so-obtained partial solutions into an overall score.

Besides the above principles, it is also sometimes convenient, in order to offer a better and more intuitive interpretation of the results, to assume that the dissimilarity between two patterns yields a *score value*, normalized in the $[0, 1]$ range (the higher the score, the higher the dissimilarity). Unless otherwise stated, we will implicitly make this assumption throughout the paper.

We start by describing how the first principle is applied to the basic case of simple patterns, after that we show how to generalize the framework to the case of complex patterns.

### 3.1. Dissimilarity between simple patterns

The dissimilarity between two patterns, $p_1$ and $p_2$, of a simple pattern type *PT* is based on three key ingredients:

- a *structure dissimilarity* function, $dis_{struct}$, that evaluates the dissimilarity of the structure components of the two patterns, $p_1.s$ and $p_2.s$,
- a *measure dissimilarity* function, $dis_{meas}$, used to assess the dissimilarity of the corresponding measure components, $p_1.m$ and $p_2.m$, and
- a *combining* function, *Comb*, also called the *combiner*, yielding an overall score from the structure and measure dissimilarity scores.

The dissimilarity of two patterns is consequently determined as (see also Fig. 1)

$$dis(p_1, p_2) = Comb(dis_{struct}(p_1.s, p_2.s), dis_{meas}(p_1.m, p_2.m)). \quad (1)$$

If $p_1$ and $p_2$ share the same structure, then $dis_{struct}(p_1.s, p_2.s) = 0$. In the general case, in which the patterns have different structures, two alternatives exist:

1. The structural components are somewhat "compatible", in which case we interpret $dis_{struct}(p_1.s, p_2.s)$ as the "additional dissimilarity" one wants to charge with respect to the case of identical structures.
2. Structures are completely unrelated (in a sense that depends on the case at hand), i.e., $dis_{struct}(p_1.s, p_2.s) = 1$. In this case, regardless of the measure dissimilarity, we also require the overall dissimilarity to be maximum, i.e., $dis(p_1, p_2) = 1$. This restriction is enforced to prevent cases where two completely different patterns might be considered somehow similar due to low differences in their measures.

**Example 6.** Continuing Example 5, consider two keywords $k_1 = (t_1, w_1)$ and $k_2 = (t_2, w_2)$ to be compared. For the structure dissimilarity function, if the two terms are the same, then $dis_{struct}(t_1, t_2) = 0$. When $t_1 \neq t_2$, if some information about the semantics of the terms is available, such as a thesaurus or a hierarchical hypernymy/hyponymy ontology, like WordNet [11], then one could set $dis_{struct}(t_1, t_2) < 1$ to reflect the "semantic distance" between $t_1$ and $t_2$ [12]; on the other hand, if no such information is available, then $dis_{struct}(t_1, t_2) = 1$. A possible choice for the measure dissimilarity function is the absolute difference of measures, i.e., $dis_{meas}(w_1, w_2) = |w_1 - w_2|$. Finally, a possible combiner for this example is, say, the *algebraic disjunction* of the two dissimilarities:

$$dis(k_1, k_2) = dis_{struct}(t_1, t_2) + dis_{meas}(w_1, w_2) - dis_{struct}(t_1, t_2) \cdot dis_{meas}(w_1, w_2), \quad (2)$$

that correctly yields $dis(k_1, k_2) = 1$ when $dis_{struct}(t_1, t_2) = 1$, and $dis(k_1, k_2) = dis_{meas}(w_1, w_2)$ when $dis_{struct}(t_1, t_2) = 0$.

**Example 7.** Continuing our other running example (Example 4), WINDSURF uses the Bhattacharyya distance [10] to compare regions, i.e., clusters structures:

$$dis_{struct}(p_1.s, p_2.s)^2 = \sum_{b=1}^{4} \left( 1/2 \cdot \ln \frac{det\left(\frac{p_1.bands[b].cov + p_2.bands[b].cov}{2}\right)}{\sqrt{det(p_1.bands[b].cov) \cdot det(p_2.bands[b].cov)}} + 1/8 \left( (p_1.bands[b].c - p_2.bands[b].c)^T \cdot \left( \frac{p_1.bands[b].cov + p_2.bands[b].cov}{2} \right)^{-1} \cdot (p_1.bands[b].c - p_2.bands[b].c) \right) \right),$$

where $det(\cdot)$ denotes the determinant of a matrix. The measure dissimilarity is defined as

$$dis_{meas}(p_1.m, p_2.m) = |p_1.supp - p_2.supp|.$$

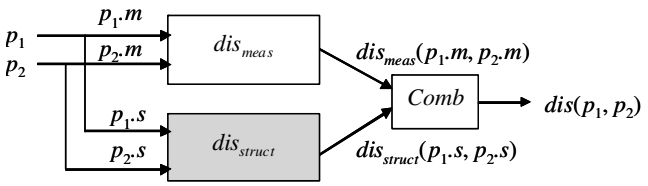


**Fig. 1.** Assessment of dissimilarity between (simple) patterns.

Finally, the combining function simply averages the two distances.

It has to be observed that, in several cases, patterns have no measure at all; for instance, sets of strings have type: $\texttt{Set} = (SS : \{\texttt{String}\}, MS : \perp)$. In this case the assessed dissimilarity will depend *only* on how much the structural components of the patterns differ, i.e., $dis(p_1, p_2) = dis_{struct}(p_1.s, p_2.s)$.

It has to be remarked that our framework does not preclude the possibility of defining different $dis_{struct}$, $dis_{meas}$ and *Comb* functions for each pattern type of interest. Rather, the functions best suited to the case at hand should be chosen, possibly depending on specific user's needs, e.g., to focus the comparison only on some patterns' properties, to trade-off accuracy for computational costs, etc. (see also Section 5).

### 3.2. Dissimilarity between complex patterns

Although in line of principle one could define simple patterns with arbitrarily complicated structural components, this would necessarily force dissimilarity functions to be complex as well and hardly reusable. Among the requirements stated in the introduction, this "monolithic" approach would only comply with that of efficiency, however, failing to address any of the other ones. In PANDA, we pursue a *modular* approach that, by definition, is better suited to guarantee flexibility, simplicity, and reusability. Moreover, as it will be discussed later, this does not rule out the possibility of efficient implementations.

Coherently with the second principle inspiring our approach, the dissimilarity of complex patterns is evaluated starting from the dissimilarities of the corresponding component patterns. In particular, the structure of complex patterns plays here a major role, since it is where pattern composition occurs.

Without loss of generality, in what follows it is assumed that the component patterns, $p^1, p^2, \ldots, p^N$, of a complex pattern $cp$ completely describe the structure of $cp$ (no additional information is present in $cp.s$) and that they form a set, i.e., $cp.s = \{p^1, p^2, \ldots, p^N\}$. At the end of this section, we describe how complex patterns built using other type constructors (lists, vectors, and tuples) can be dealt with.

The structure dissimilarity of complex patterns $cp_1.s = \{p_1^1, p_1^2, \ldots, p_1^{N_1}\}$ and $cp_2.s = \{p_2^1, p_2^2, \ldots, p_2^{N_2}\}$ depends on two fundamental abstractions, namely:

- the *matching type*, which is used to establish how the component patterns of $cp_1$ and $cp_2$ can be *matched*, and
- the *aggregation logic*, which is used to combine the dissimilarity scores of the matched component patterns into a single value representing the total dissimilarity between the structures of the complex patterns.

#### 3.2.1. Matching type

A *matching* between the complex patterns $cp_1.s = \{p_1^1, p_1^2, \ldots, p_1^{N_1}\}$ and $cp_2.s = \{p_2^1, p_2^2, \ldots, p_2^{N_2}\}$ is a matrix $\mathbf{X}_{N_1 \times N_2} = (x_{i,j})_{ij}$, where each element $x_{i,j} \in [0,1]$ $(i = 1, \ldots, N_1;\ j = 1, \ldots, N_2)$ represents the (amount of) matching between the $i$th component pattern of $cp_1$ and the $j$th component pattern of $cp_2$, i.e., between $p_1^i$ and $p_2^j$.

A *matching type* is a set of constraints on the $x_{i,j}$ coefficients so that only some matchings are valid. Relevant cases of matching types include:

1–1 *matching:* In this case, each component pattern of $cp_1$ (resp., $cp_2$) might be matched to at most one component pattern of $cp_2$ (resp., $cp_1$). Partial matching occurs if $N_1 \neq N_2$. The 1–1 matching type corresponds to the following set of constraints:

$$\sum_{i=1}^{N_1} x_{i,j} \leqslant 1 \quad \forall j, \qquad \sum_{j=1}^{N_2} x_{i,j} \leqslant 1 \quad \forall i \qquad \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} x_{i,j} = \min\{N_1, N_2\}, \quad x_{i,j} \in \{0,1\}\ \ \forall i,j$$

*N—M (complete) matching:* In this case, each component pattern of $cp_1$ (resp., $cp_2$) is matched to every component pattern of $cp_2$ and vice versa, i.e., $x_{i,j} = 1, \forall i,j$.

*EMD matching:* This matching type, introduced for defining the earth mover's distance (EMD) [13,14], differs from previous ones in that each $x_{i,j}$ might be real-valued, and represents the amount of $p_1^i$ "mass" that is matched with $p_2^j$. The corresponding constraints on the matching matrix are:

$$\sum_{i=1}^{N_1} x_{i,j} \leqslant w_2^j \quad \forall j \qquad \sum_{j=1}^{N_2} x_{i,j} \leqslant w_1^i \quad \forall i, \qquad \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} x_{i,j} = \min\left\{\sum_{i=1}^{N_1} w_1^i, \sum_{j=1}^{N_2} w_2^j\right\}, \quad x_{i,j} \in [0,1]\ \ \forall i,j,$$

where $w_1^i$ (resp., $w_2^j$) is the weight (mass amount) associated to each component pattern $p_1^i$ (resp., $p_2^j$) of $cp_1$ (resp., $cp_2$).

Finally, note that dissimilarity functions rely, either explicitly or implicitly, on a specific matching type. For instance, the *N—M* complete matching is used by the complete linkage algorithm to compare clusters. Variations of matching types described above are also common, such as the one used by the dynamic time warping (DTW) distance [15] as well as related distances for time series (see also Section 4.3).

*3.2.2. Aggregation*

For computing the total dissimilarity between complex patterns, the dissimilarity scores of matched component patterns have to be *aggregated* so as to obtain a single score. In general, this is represented by an *Aggr* function, which takes as input the matrix $\mathbf{D}_{N_1 \times N_2} = (dis(p_1^i, p_2^j))_{ij}$ of dissimilarities between pattern components and a matching matrix:

$$Aggr(\mathbf{D}, \mathbf{X}),$$

where the above usually takes the form:

$$Aggr((dis(p_1^i, p_2^j) \times x_{i,j})_{ij}),$$

i.e., the $x_{i,j}$ coefficients are used to weigh components' dissimilarities (this is the case for all the examples in this paper).

Among all the valid matchings (as specified by the matching type), the rationale is to pick the "best" one, i.e., the one that minimizes the aggregated structure dissimilarity:

$$dis_{struct}(cp_1.s, cp_2.s) = \min_{\mathbf{X}}\{Aggr(\mathbf{D}, \mathbf{X})\}. \tag{3}$$

Putting all together, the computation of structure dissimilarity of complex patterns can be summarized as in Fig. 2:

- The "*Matcher*" block is responsible for specifying which are the valid matchings, i.e., those respecting the constraints of the chosen matching type.
- The "*Aggregator*" block, which states how scores of the matched component pairs should be aggregated so as to yield the overall score.
- The "*Optimizer*" block is in charge of determining the best matching, i.e., the one with the lowest dissimilarity score. Note that, in the general case, the logic of the optimizer depends on both the matching type and the aggregator chosen for the case at hand (see also Section 5).
- In case of multi-level aggregations, the dissimilarity block (gray box in Fig. 2) might encompass the recursive computation of dissimilarity between complex patterns.

Finally, the overall dissimilarity between $cp_1$ and $cp_2$ is as in Eq. (1), thus following the same approach as with simple patterns.

**Example 8.** To complete Example 4, for comparing images WINDSURF has to match their regions. To this end, WINDSURF provides two different modalities: the first one uses the 1–1 matching type and an aggregation function defined as the average over the matched regions:

$$Aggr_{avg} = \frac{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} dis(p_1^i, p_2^j) \times x_{i,j}}{\min\{N_1, N_2\}}.$$

The second modality applies EMD matching, and the aggregator:

$$Aggr_{\mathrm{EMD}} = \frac{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} dis(p_1^i, p_2^j) \times x_{ij}}{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} x_{i,j}},$$

where the mass amount (weight) of each cluster equals its support. Note that, since regions' supports are explicitly used by the EMD matching, they are not to be used when comparing regions; it follows that the combiner at the region level only takes into account the structure dissimilarity, i.e., $dis(p_1^i, p_2^j) = dis_{struct}(p_1^i.s, p_2^j.s)$.

**Example 9.** Completing Example 5, we first show how two documents (complex patterns modeled as sets of weighted keywords) can be compared. For simplicity, we only consider the case where the structure dissimilarity between keywords yields a binary value, thus, according to Eq. (2), $dis(k_1, k_2) = |w_1 - w_2|$, if $t_1 = t_2$, and $dis(k_1, k_2) = 1$, otherwise. The distance

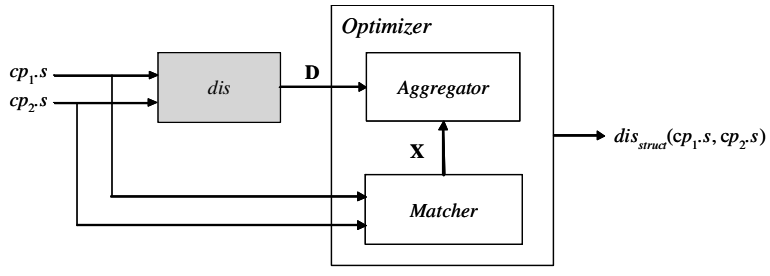


**Fig. 2.** Assessment of structure dissimilarity between complex patterns.

between documents considers differences in weight between matched (thus, equal) keywords plus a penalty value for unmatched keywords. This can be represented, in our framework, by taking into account an 1–1 matching and averaging over the number of distinct keywords in the two patterns:

$$Aggr = \frac{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} dis(p_1^i, p_2^j) \times x_{i,j}}{|p_1.s \cup p_2.s|}.$$

Finally, collections of documents are modeled as sets of documents and the 1–1 matching and *avg* aggregator can be considered, for example.

When the structure of a complex pattern consists of a list (or a vector) of component patterns (instead of a set), it is quite common to forbid "inversions" in the matching, i.e., if $p_1^i$ is matched with $p_2^j$, then $p_1^{i'}$ cannot be matched with $p_2^{j'}$, with $i' > i$ and $j' < j$ (see also Section 4.3): this results in additional constraints to the generation of valid matchings. The case of complex patterns consisting of tuples of component patterns is even simpler, since only corresponding components can be matched.

## 4. Implementing specific dissimilarity measures

In order to show the generality of the PANDA framework, in this section we describe how it can model specific dissimilarity measures that have been proposed in the literature for some DM tasks.

### 4.1. Clusterings comparison

Among the many approaches available to compare clusterings obtained from a same dataset using different algorithms or parameters (see [16] for a survey), the one based on *variation of information* [17] has to be appreciated for being grounded on information-theoretical principles.

The variation of information measure considers the *entropies*, $H(cp_1)$ and $H(cp_2)$, of the two clusterings, $cp_1$ and $cp_2$, as well as their *mutual information*, $I(cp_1, cp_2)$, that is

$$d_{VI}(cp_1, cp_2) = H(cp_1) + H(cp_2) - 2I(cp_1, cp_2),$$

where $H(cp) = \sum_j H(p^j)$ and $I(cp_1, cp_2) = \sum_i \sum_j I(p_1^i, p_2^j)$. In turn, the mutual information of two clusters, each represented as a simple pattern ($s = S, m = \perp$), $S$ being the set of points in the cluster, is defined as

$$I(p_1^i, p_2^j) = \frac{\left|p_1^i.s \cap p_2^j.s\right|}{n} \log \frac{n \cdot \left|p_1^i.s \cap p_2^j.s\right|}{\left|p_1^i.s\right|\left|p_2^j.s\right|},$$

where $n$ is the total number of points in the dataset. To see how this fits our framework, define the distance between clusters $p^i$ and $p^j$ as

$$dis(p_1^i, p_2^j) = dis_{struct}(p_1^i.s, p_2^j.s) = \frac{H(p_1^i)}{N_2} + \frac{H(p_2^j)}{N_1} - 2I(p_1^i, p_2^j)$$

and consider an $N$—$M$ (complete) matching type. Summing over all the possible pairs of clusters, it is obtained $d_{VI}(cp_1, cp_2) \equiv dis(cp_1, cp_2) = \sum_i \sum_j dis(p_1^i, p_2^j)$.[3]

### 4.2. Frequent itemsets comparison

Frequent itemset mining is one of the core tasks in DM, aiming at uncovering relationships between data items. Given a set of items $I$, an *itemset* is any non-empty subset of $I$. A *transaction* is a set of items as well. Given a set of transactions $T$, the *support* of an itemset $x$ in $T$, $supp_T(x)$, is the fraction of transactions in $T$ containing $x$. Given a minimum support value, $supp_{\min}$, $x$ is a *frequent itemset* iff $supp_T(x) \geqslant supp_{\min}$. Several algorithms exist for efficiently computing the set of frequent itemsets from a dataset $T$, including the well-known Apriori [18] algorithm.

The similarity measure proposed in [5] for comparing two sets $A$ and $B$ of frequent itemsets, derived from dataset $T_A$ and $T_B$, respectively, is

$$sim(A, B) = \frac{\sum_{x \in A \cap B} \max\left\{0, 1 - \alpha \cdot \left|supp_{T_A}(x) - supp_{T_B}(x)\right|\right\}}{|A \cup B|},$$

where $\alpha$ is an user-specified scaling parameter indicating the significance of the difference in support. This similarity measure can be converted into an equivalent distance measure fitting our framework as follows. A frequent itemset is a simple

[3] To stay in line with the original formulation of $d_{VI}$ in [17], we have not normalized the dissimilarity score.

pattern ($s = S, m = supp$), where $S$ is the set of items (structural component) and $supp \in [0,1]$ is the support of the itemset (measure component). The distance between itemsets $p^i$ and $p^j$ is defined as

$$dis_{struct}(p^i.s, p^j.s) = \begin{cases} 0 & \text{if } p^i.s = p^j.s, \\ 1 & \text{otherwise}, \end{cases}$$
$$dis_{meas}(p^i.m, p^j.m) = \min\{\alpha \cdot |p^i.m - p^j.m|, 1\},$$
$$dis(p^i, p^j) = 1 - (1 - dis_{struct}(p^i.s, p^j.s)) \cdot (1 - dis_{meas}.(p^i.m, p^j.m)).$$

To compare two sets of itemsets, $cp_1$ and $cp_2$, an $N$–$M$ matching is considered, while the aggregation function is a simple average:

$$dis(cp_1, cp_2) = dis_{struct}(cp_1.s, cp_2.s) = 1 - \frac{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}(1 - dis(p_1^i, p_2^j))}{|cp_1.s \cup cp_2.s|},$$

that is, $dis(cp_1, cp_2) = 1 - sim(cp_1, cp_2)$.

### 4.3. Time series comparison

A time series is an ordered sequence of samples, where each sample value depends on the domain at hand, e.g., a real (the stock value for a particular day) for stock market data, a two-dimensional point for trajectories/shapes, etc.

One of the most common measures for assessing the dissimilarity among time series is the dynamic time warping (DTW) distance [15], that allows both the comparison of series with different length and the matching of samples at different time-stamps. The computation of the DTW distance between two sequences is based on the concept of *warping path*. Given two sequences, $cp_1 = \langle cp_1[1], \ldots, cp_1[N_1]\rangle$ and $cp_2 = \langle cp_2[1], \ldots, cp_2[N_2]\rangle$, and a base distance *dis* used to compare samples, one can compute the $\mathbf{D}_{N_1 \times N_2} = (dis(cp_1[i], cp_2[j]))_{ij}$ matrix containing pairwise distances of the two sequences; a warping path $\mathbf{W}$ is then a contiguous set of elements of $\mathbf{D}$, $w_k = dis(cp_1[i_k], cp_2[j_k])$, that satisfies the following constraints:

*Boundary conditions:* The first and the last samples of the two sequences should be matched together, i.e., $i_1 = j_1 = 1$ and, being $K$ the length of $W$, $i_K = N_1, j_K = N_2$.
*Continuity and monotonicity:* Steps in the warping path should always proceed forward in time and match adjacent cells, i.e., $0 \leqslant i_k - i_{k-1} \leqslant 1$ and $0 \leqslant j_k - j_{k-1} \leqslant 1$.

Each warping path has an associated cost that can be obtained by summing all its elements. The DTW distance is defined as the minimum cost between all the warping paths:

$$\mathrm{DTW}(p, q) = \min_W \sum_{k=1}^{K} w_k.$$

Note that, although an exponential number of warping paths exists, the DTW can be efficiently computed in $\mathcal{O}(N_1 \times N_2)$ time using dynamic programming.

The DTW distance can be easily modeled within the PANDA framework by considering each sequence as a complex pattern whose structure is a vector of samples (simple patterns). Clearly, each warping path defines a way to match elements of the two sequences; the constraints on the paths can be translated, using our formalism, as

$$x_{i,j} \in \{0,1\}, \quad x_{1,1} = 1, \quad x_{N_1,N_2} = 1, \quad x_{i,j} = 1 \Rightarrow x_{i-1,j} + x_{i,j-1} + x_{i-1,j-1} = 1$$

and the DTW distance is then defined as (again, the dissimilarity is not normalized)

$$\mathrm{DTW}(cp_1, cp_2) \equiv dis(cp_1, cp_2) = dis_{struct}(cp_1.s, cp_2.s) = \min_{\mathbf{x}} \left\{ \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} dis(cp_1[i], cp_2[j]) \cdot x_{i,j} \right\}.$$

## 5. The software framework

The PANDA conceptual framework is amenable to be implemented as a software framework, which we did by coding a number of basic building blocks in Java.[4] These blocks include all the components of the conceptual framework, i.e., structure and measure dissimilarities, combiners for the simple patterns case, as well as a set of optimizers for specific matching types. The software framework is designed so that dissimilarity computation tasks can be performed in a modular way, coherently with the conceptual framework.

[4] The code is freely available for non-commercial use and also includes a sample application for the comparison of different kinds of patterns, like clusterings and time series. Interested readers, please contact the authors for information on how to download it.

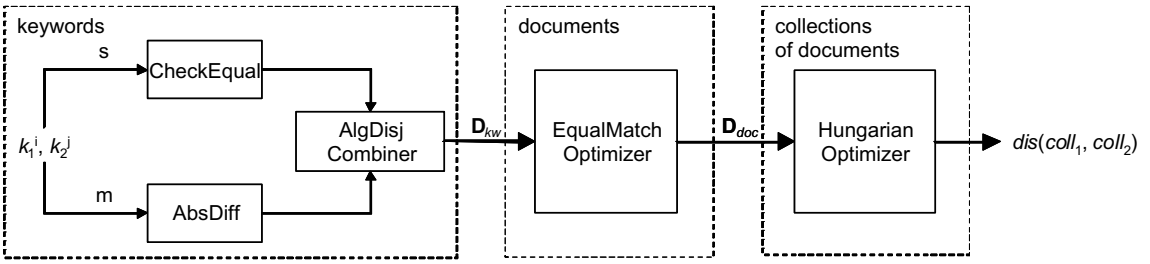


**Fig. 3.** Software components needed to compare two collections of documents.

As an example, Fig. 3 shows the software components needed to evaluate the dissimilarity of two collections of documents. As discussed in Example 9, this first requires to compare simple keywords (checking whether keywords are the same, then using the absolute difference of weights), then to match keywords (this is performed by the EqualMatch Optimizer that only matches keywords common to two documents), and finally to match documents. Note that for the latter an assignment problem has to be solved, thus an optimizer implementing the Hungarian algorithm [19] is required.

From both implementative and computational points of view, the optimizer is usually the most complex component in the chain leading to determine the overall dissimilarity score. Since in many cases the number of valid matchings grows exponentially with the number of component patterns (e.g., this is the case for the EMD distance, the 1–1 matching type, and the DTW distance), any non-trivial implementation needs to avoid enumerating all of them. The Hungarian optimizer, which runs in $\mathcal{O}(N^3)$ time, as well as the optimizer needed to compute the EMD distance (whose complexity is $\mathcal{O}(N^3 \log N)$), can however represent a performance bottleneck in environments where the user is willing to use our framework to explore alternative definitions of pattern dissimilarities. To this end, a simple solution is to (slightly) deviate from the definition of dissimilarity as the minimum among all valid matchings and also consider sub-optimal (i.e., approximate) optimizers. For instance, we implemented both an Hungarian optimizer as well as a greedy optimizer for the 1–1 matching type, the latter typically providing one order of magnitude speed-up over the former for patterns with tens–hundreds of components and often leading to similar results (see Appendix A.1).

Adopting the PANDA framework at the software level has the neat advantage of immediately favoring the reusability of implemented software components, since each of them has a well-defined functionality. For the same reason, PANDA makes easier to develop libraries of user-defined building blocks that can be freely combined so as to provide a rich set of ready-to-use alternatives for dissimilarity assessment.

## 6. Related work

In this section, we compare our approach with other proposals related to pattern management, with the aim of making clear the uniqueness of PANDA objectives. In Section 6.1, we contrast PANDA to FOCUS [4], which to the best of our knowledge is the only other framework aiming to a principled approach for comparing patterns.

Works on *inductive databases* (IDBs) [20] stress the need for storing together data and patterns so as to uniformly retrieve and manipulate them. The emphasis in IDBs has traditionally been on pattern querying and retrieval, leading to the development of specialized query languages [21,22]. On the other hand, pattern comparison *per se* is not a major issue in IDBs. This is also because IDBs only consider specific types of patterns, like association rules and string patterns, and do not deal with complex patterns at all.

*Pattern monitoring* aims at observing the development of patterns over time and detecting their changes. Relevant solutions for pattern monitoring are DEMON [23] and PAM [24]. Clearly, pattern monitoring builds upon pattern comparison in order to detect changes across the temporal dimension; however, apart from this, emphasis is on completely different issues, e.g., efficient update of the pattern base, assessment of the statistical significance of changes, etc.

*Kernel methods* are becoming quite popular in the DM community, since they can be applied to perform machine learning tasks on complex data without the burden of running algorithms on high-dimensional data [25]. To this end, the dataset is embedded in a linear space so that the kernel on any elements of the set corresponds to the inner product in such a space. Somehow related to our work is the research on convolution kernels for structured data [26], whose basic idea is to capture the semantics of composite/complex objects by means of a relationship among the object itself and its components. In this way, the kernel on the complex object is obtained from the kernels defined on its parts. It is apparent that this approach shares with ours the basic idea of recursive decomposition of complex patterns, the substantial difference being that PANDA focuses on comparing the results of DM processes, whereas kernel methods are tools for supporting such processes. An interesting research issue would be to exploit work on kernels to enhance our framework along the direction of *dissimilarity reasoning*, i.e., how to relate dissimilarity in data and pattern space.

Two notions of similarity among patterns are defined in [3]:

- The *explicit similarity* of two patterns is computed as the (normalized) intersection of their data members, i.e., of the data the two patterns where extracted from; clearly, when two patterns come from different datasets, their similarity equals to

zero. Moreover, assessing the similarity between two patterns requires accessing the underlying datasets, thus one of the main requisites for an efficient pattern comparison (see Section 1) is violated.
- The *implicit similarity* of two patterns is defined as the volume of the intersection of the regions corresponding to the patterns themselves. Obviously, this requires that patterns are defined over some vector space, i.e., $SS = \Re^n$. The generality of such approach is clearly very limited, since several pattern types are not defined over a vector space, e.g., frequent itemsets; moreover, this notion of similarity cannot be directly applied to complex patterns.

### 6.1. The FOCUS framework

FOCUS [4] provides a principled approach for measuring the deviation between two datasets, in terms of corresponding extracted patterns. This has been demonstrated on three popular pattern types, namely frequent itemsets, decision trees and non-overlapping clusterings.

The central idea of FOCUS, which has also been a major inspiration to the design of PANDA, is to view patterns as made up by structure and measure components. In FOCUS a pattern is modeled as a set of "regions" over the attribute space (the pattern structure) to each of which a set of measures is associated. For instance, a decision tree over $n$ attributes and classifying tuples into $m$ classes partitions the attribute space into a set of regions. Each region corresponds to a leaf node of the tree and has $m$ measures, one for each class, representing the fraction of tuples in that region that actually belong to that class.

For measuring the dissimilarity of two patterns, structural components, i.e., regions, have to be first "reconciled" by minimally refining/splitting them until the two structures become identical; the result is called the greatest common refinement (GCR) of the two patterns. After that, the overall dissimilarity is computed by aggregating over all regions differences among corresponding measures. For instance, the GCR of two decision trees is the decision tree whose regions are obtained by intersecting the sets of regions of the two trees. In general, this requires computing from the underlying datasets measure values for the (new) regions in the GCR.

The comparison of FOCUS with PANDA reveals a number of limitations that prevent using the former as a general and flexible framework for pattern comparison:

- FOCUS only considers patterns whose structure consists of a two-levels hierarchy, i.e., each pattern is composed of regions, while PANDA supports the recursive definition of arbitrarily complex patterns. A number of interesting pattern types are therefore left out by FOCUS. On the other hand, all the patterns types that are supported by FOCUS can be also managed by PANDA.
- Because of the use of the *GCR*, FOCUS is not flexible enough concerning the matching of component patterns. This prevents comparing patterns based on some "holistic" criterion, since FOCUS only considers a simple aggregation of differences of measures on corresponding regions. As also shown by examples in Section 4, this kind of 1–1 matching type is not general enough.
- The use of the *GCR* also prevents the applicability of FOCUS to a number of interesting patterns for which the notion of common refinement cannot be defined. For example, several examples used in this paper cannot be modeled by FOCUS.
- Since FOCUS is concerned with the comparison of datasets, it requires a comparison of the raw data underlying the patterns, whereas pattern comparison in PANDA is totally processed in the pattern space, thus an improvement is obtained in efficiency (because costly accesses to the raw data are avoided), generality (we can deal with cases where raw data are not available), and privacy (there is now no need to access the data that may contain private information).

## 7. Conclusions

In this paper, we have presented the PANDA framework for the comparison of patterns. The general model of PANDA is based on the definition of patterns as two-components (structure and measure) entities and on the recursive definition of complex patterns. The process of comparing patterns is built on a few basic notions in order to be easily applicable and understandable. In case of simple patterns, these notions are: the *structure* and *measure dissimilarity* defined over the corresponding components of patterns, and the *combining function*, which defines how these scores are combined to produce a single *dissimilarity score*. In case of complex patterns, additional notions are required, namely: the *recursive* definition of dissimilarity in terms of their component (simple or complex) patterns, the *matching type*, which defines how the component patterns are matched, and the *aggregation logic*, which aggregates the scores of the matched component patterns into an overall dissimilarity score. The above mentioned notions constitute the building blocks of the PANDA framework, their instantiations resulting in (possibly different) dissimilarity functions configurations.

A number of interesting working issues are still left open. First, it would be interesting to understand if PANDA principles could be somehow exploited to derive a complementary *indexing framework* for the efficient evaluation of dissimilarity queries over large pattern bases. Although for specific dissimilarity measures one can adopt specific indexing solutions, a thing we did by using *M*-tree [27,28] for some of the examples discussed in this paper, a general solution is still missing. A second relevant issue would be to study the relationship between the dissimilarity computed over patterns and the one computed over the underlying dataset(s). As observed in Section 6, work on kernel methods could be exploited to this end.

## Appendix A

### Experiments using the PANDA framework

We have applied PANDA to three DM experimental scenarios, with the aim of (a) demonstrating its applicability to a wide range of applications (generality) and (b) presenting how PANDA can be easily adapted to the peculiarities of each case (flexibility).

#### A.1. Sets of itemsets

In this experiment, we use the synthetic dataset generator from the IBM Quest data mining group [29], which is assumed to mimic the transactions in a retailing environment, to generate a dataset $\mathscr{D}$ of 1000 transactions with an average transaction length of 10 attributes. For the extraction of frequent itemsets we use MAFIA [30] with a *minSupport* threshold of 20%.

Itemsets and sets of itemsets are modeled as in Section 4.2; however, differently from [5], we use an 1–1 matching type to match itemsets and the average as the aggregation function. To compute the optimal match, we use both an exact Hungarian optimizer and a greedy one, which at each step pairs together the least dissimilar unmatched patterns. Fig. 4 shows the software components needed to run the experiment.

With the aim of investigating how differences in underlying data influence the dissimilarity of the corresponding patterns, in the experiment we start from a dataset $\mathscr{D}_0$, generated as above described and whose set of frequent itemsets is $P_0$. Then, we compare $P_0$ with patterns $P_x$ ($x = 10, 20, \ldots, 100$) extracted from some noisy versions $\mathscr{D}_x$ of $\mathscr{D}_0$, where $\mathscr{D}_x$ is obtained by modifying $x\%$ of the transactions in $\mathscr{D}_0$, for each of them changing 50% of its items.

As illustrated in Fig. 5, the dissimilarity at the pattern level reflects the increased dissimilarity between the datasets when the amount of noise grows. Further, since results from the two optimizers are very similar, a practical conclusion of this experiment is that the greedy optimizer can be safely chosen for the comparison of sets of frequent itemsets, since it yields much faster response times (25$x$ on average, in our case).

#### A.2. Clusterings (images)

Our second experiment deals with the comparison of images (see Example 4). In order to show how differences at the data (image) level are reflected by the dissimilarity score computed at the pattern level, we consider a set of more than 2000 images (the same data used in [10,31]) and modify each image by introducing white noise. In detail, we add to the RGB components of each image pixel a Gaussian value with mean 0 and standard deviation $\sigma$ in the interval [0,50]. Fig. 6 shows a visual example. We then compare each image with its noisy versions by using our framework, implemented as in Fig. 7.

Average results are plotted in Fig. 8. As the graph clearly shows, the dissimilarity computed at the pattern level is indeed able to capture the increasing dissimilarity between images when the amount of added noise at the raw data level increases. As to the comparison of the Hungarian and the EMD optimizers, Fig. 8 shows that EMD is more sensitive to noise, although at

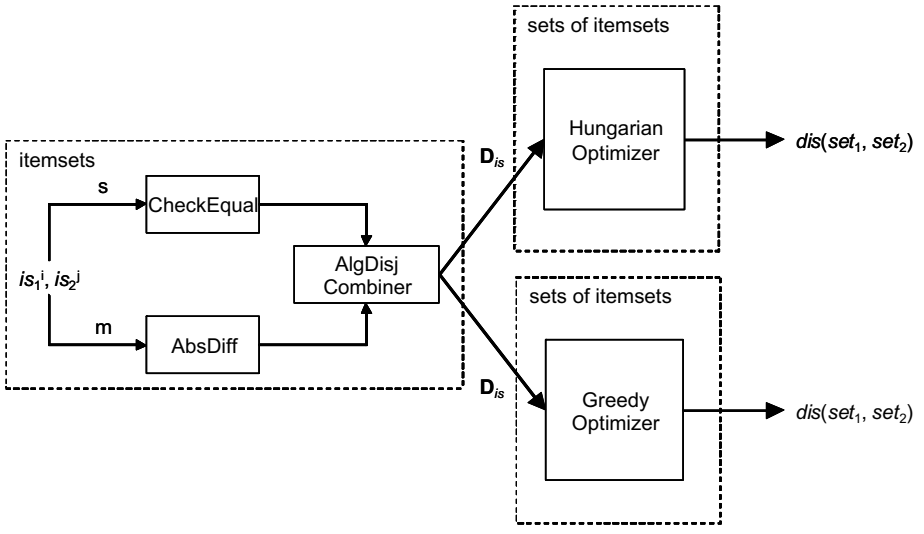


**Fig. 4.** Software components for comparing sets of itemsets.

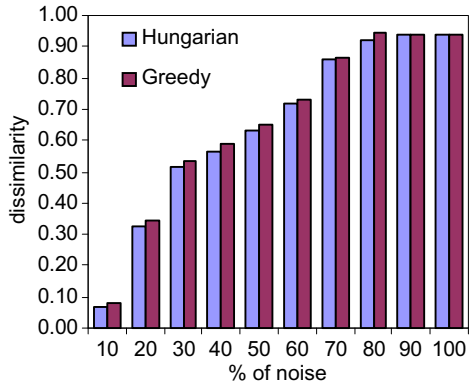


**Fig. 5.** Impact of dataset noise on the dissimilarity of sets of itemsets for the Hungarian and the Greedy optimizers.


**Fig. 6.** The original image (a) and noisy images: $\sigma = 10$ (b), $\sigma = 50$ (c).

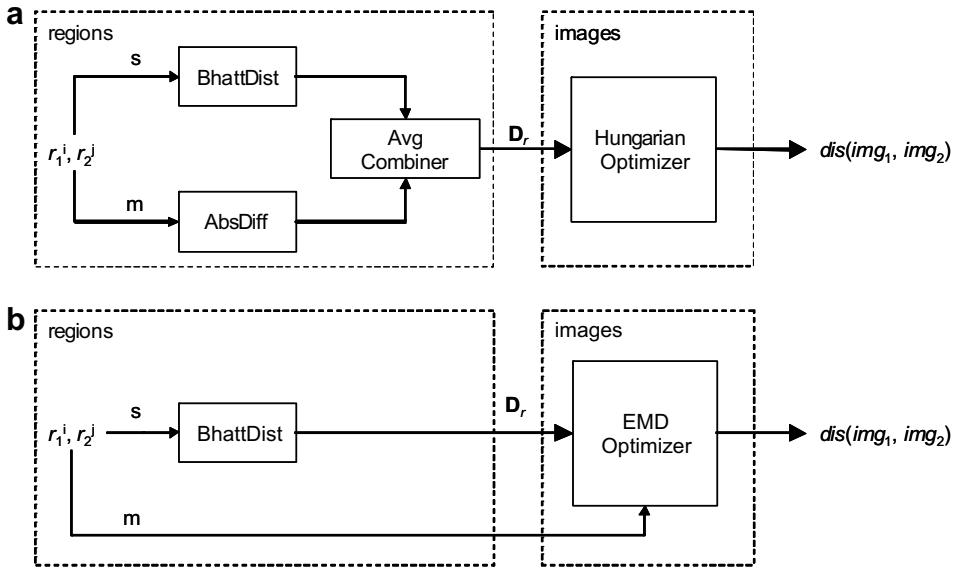


**Fig. 7.** Software components for comparing images: (a) 1–1 matching, (b) EMD matching.

low noise levels (<15%) the average EMD dissimilarity is lower than the one obtained with the Hungarian optimizer. Concerning execution times, both methods can compute the dissimilarity of two images in less than 1 ms.

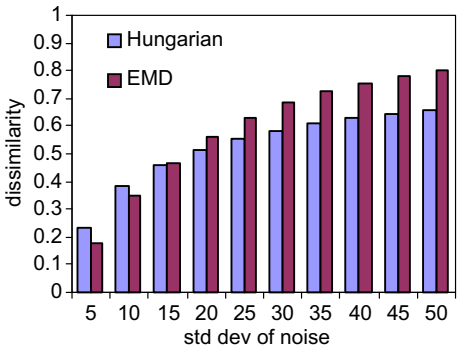


**Fig. 8.** Impact of noise on images dissimilarity for the Hungarian and the EMD optimizers.

### *A.3. Collections of documents*

Our final experiment refers to the comparison of collections of documents (see Example 5). Keywords are compared as in Example 6, while complex patterns are compared as in Example 9 (software components for this experiment were illustrated in Fig. 3).

**Table 1**
DBLP journals.

| DBLP journal | Abbreviation | DBLP journal | Abbreviation |
|---|---|---|---|
| Computer Journal | Comp J | Computational Intelligence | Comp Intell |
| Artificial Intelligence | AI | Distributed Computing | Dist Comp |
| Computer Networks | Comp N | Advances in Computers | Adv in Comp |
| Computers and Graphics | Comp G | Evolutionary Computation | Evol Comp |
| Information Systems | Info Sys | Computational Complexity | Comp Compl |
| Information Retrieval | IR | Comp. Networks and ISDN Syst. | Comp Net & ISDN Sys |
| Computer Languages | Comp Lang | | |

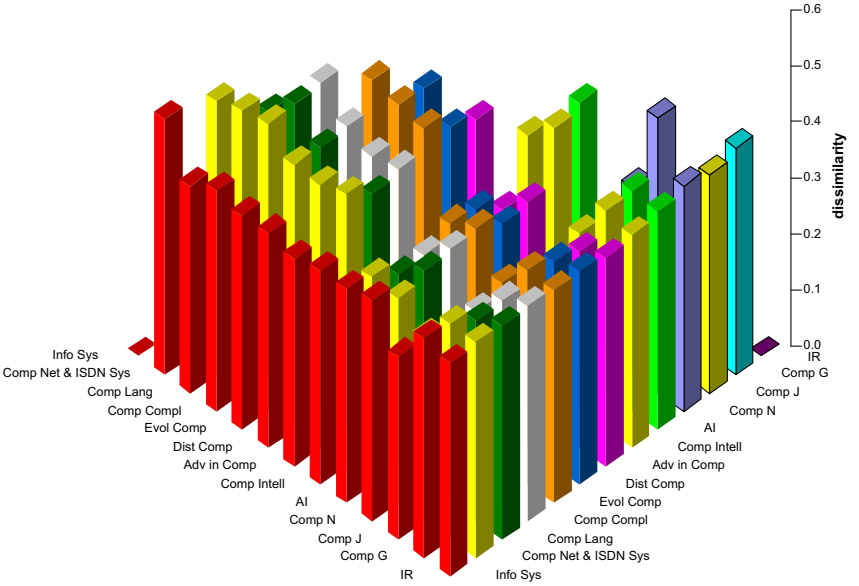


**Fig. 9.** Comparing journals in the DBLP database.

We consider a set of journals from the DBLP [32] database, each journal containing a set of articles referring to a same topic (the one covered by the journal). Journals used in this experiment are listed in Table 1.

The results of the experiment are shown in Fig. 9, allowing us to draw the following interesting conclusions:

- The *Computer Journal* is quite similar to all other journals, a result that is reasonable since this journal provides a complete overview of developments in the broad field of computer science.
- On the other hand, journals like *Evolutionary Computation*, *Computational Intelligence*, *Computer Networks*, *ISDN Systems*, and *Distributed Computing* are very dissimilar to each other, since they cover specific and rather distinct topics in computer science.
- The greatest distance found was the one between the *Evolutionary Computation* and the *Distributed Computing* journals, which cover totally distinct topics.

## References


[1] P. Lyman, H.R. Varian, How much information, 2003. <http://www.sims.berkeley.edu/research/projects/how-much-info-2003> (valid as of October 3, 2008).
[2] Y. Theodoridis, M. Vazirgiannis, P. Vassiliadis, B. Catania, S. Rizzi, A manifesto for pattern bases, Tech. Rep. PANDA TR-2003-03, IST-2001-33058 Thematic Network PANDA, 2003. <http://www.pbms.org/papers/TR-2003-03.pdf> (valid as of October 3, 2008).
[3] M. Terrovitis, P. Vassiliadis, S. Skiadopoulos, E. Bertino, B. Catania, A. Maddalena, S. Rizzi, Modeling and language support for the management of pattern-bases, Data and Knowledge Engineering 62 (2) (2007) 368–397.
[4] V. Ganti, J. Gehrke, R. Ramakrishnan, A framework for measuring changes in data characteristics, in: Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'99), ACM Press, Philadephia, PA, 1999, pp. 126–137.
[5] S. Parthasarathy, M. Ogihara, Clustering distributed homogeneous datasets, in: Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2000), Lecture Notes in Computer Science, No. 1910, Springer, Lyon, France, 2000, pp. 566–574.
[6] I. Ntoutsi, N. Pelekis, Y. Theodoridis, Pattern comparison in data mining: a survey, in: D. Taniar (Ed.), Research and Trends in Data Mining Technologies and Applications: Advances in Data Warehousing and Mining, Idea Group, 2007 (Chapter 4).
[7] I. Bartolini, P. Ciaccia, M. Patella, A framework for the comparison of complex patterns, in: Proceedings of the International Workshop on Pattern Representation and Management (PaRMa 2004), CEUR Workshop Proceedings, vol. 96, CEUR-WS.org, Heraklion, Hellas, 2004.
[8] I. Bartolini, P. Ciaccia, I. Ntoutsi, M. Patella, Y. Theodoridis, A unified and flexible framework for comparing simple and complex patterns, in: Proceedings of the Eighth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004), Lecture Notes in Computer Science, No. 3202, Springer, Pisa, Italy, 2004, pp. 496–499.
[9] S. Rizzi, E. Bertino, B. Catania, M. Golfarelli, M. Halkidi, M. Terrovitis, P. Vassiliadis, M. Vazirgiannis, E. Vrachnos, Towards a logical model for patterns, in: Proceedings of the 22nd International Conference on Conceptual Modeling (ER 2003), Lecture Notes in Computer Science, No. 2813, Springer, Chicago, IL, 2003, pp. 77–90.
[10] S. Ardizzoni, I. Bartolini, M. Patella, Windsurf: region-based image retrieval using wavelets, in: Proceedings of the First International Workshop on Similarity Search (IWOSS'99), IEEE Computer Society, Florence, Italy, 1999, pp. 167–173.
[11] G.A. Miller, WordNets: a lexical database for English, Communications of ACM 38 (11) (1995) 39–41. <http://wordnet.princeton.edu> (valid as of October 3, 2008).
[12] P. Ganesan, H. Garcia-Molina, J. Widom, Exploiting hierarchical domain structure to compute similarity, ACM Transactions on Information Systems 21 (1) (2003) 64–93.
[13] Y. Rubner, C. Tomasi, L.J. Guibas, A metric for distributions with applications to image databases, in: Proceedings of the Sixth International Conference on Computer Vision ICCV'98, IEEE Computer Society, Mumbai, India, 1998, pp. 59–66.
[14] Y. Rubner, C. Tomasi, Perceptual Metrics for Image Database Navigation, Kluwer Academic Publishers, Boston, MA, 2000.
[15] E. Keogh, Exact indexing of dynamic time warping, in: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02), Morgan Kaufman, Hong Kong, China, 2002, pp. 406–417.
[16] M. Meilă, Comparing clusterings, Tech. Rep. 418, University of Washington, Department of Statistics, 2002.
[17] M. Meilă, Comparing clusterings by the variation of information, in: Proceedings of the 16th Annual Conference on Computational Learning Theory (COLT 2003), Lecture Notes in Computer Science, No. 2777, Springer, Washington, DC, 2003, pp. 173–187.
[18] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94), Morgan Kaufman, Santiago de Chile, Chile, 1994, pp. 487–499.
[19] H.W. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistic Quarterly 2 (1955) 83–97.
[20] L. De Raedt, A perspective on inductive databases, SIGKDD Explorations 4 (2) (2002) 69–77.
[21] J. Han, Y. Fu, W. Wang, K. Koperski, O. Zaiane, DMQL: a data mining query language for relational databases, in: SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96), Montreal, QC, 1996.
[22] T. Imielinski, A. Virmani, MSQL: a query language for database mining, Data Mining and Knowledge Discovery 3 (4) (1999) 373–408.
[23] V. Ganti, J. Gehrke, R. Ramakrishnan, DEMON: mining and monitoring evolving data, in: Proceedings of the 16th International Conference on Data Engineering (ICDE 2000), IEEE Computer Society, San Diego, CA, 2000, pp. 439–448.
[24] S. Baron, M. Spiliopoulou, O. Günther, Efficient monitoring of patterns in data mining environments, in: Proceedings of the Seventh East European Conference on Databases and Information Systems (ADBIS 2003), Lecture Notes in Computer Science, No. 2798, Springer, Dresden, Germany, 2003, pp. 253–265.
[25] T. Gärtner, A survey of kernels for structured data, SIGKDD Explorations 5 (1) (2003) 49–58.
[26] D. Haussler, Convolution kernels on discrete structures, Tech. Rep. UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz, July 1999.
[27] P. Ciaccia, M. Patella, P. Zezula, *M*-tree: an efficient access method for similarity search in metric spaces, in: Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97), Morgan Kaufman, Athens, Greece, 1997, pp. 426–435.
[28] P. Ciaccia, M. Patella, Searching in metric spaces with user-defined and approximate distances, ACM Transactions on Database Systems 27 (4) (2002) 398–437.
[29] R. Agrawal, M. Mehta, J.C. Shafer, R. Srikant, A. Arning, T. Bollinger, The quest data mining system, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96), AAAI Press, Portland, OR, 1996, pp. 244–249.
[30] D. Burdick, M. Calimlim, J. Gehrke, MAFIA: a maximal frequent itemset algorithm for transactional databases, in: Proceedings of the 17th International Conference on Data Engineering (ICDE 2001), IEEE Computer Society, Heidelberg, Germany, 2001, pp. 443–452. <http://himalaya-tools.sourceforge.net/Mafia/> (valid as of October 3, 2008).

[31] I. Bartolini, P. Ciaccia, M. Patella, A sound algorithm for region-based image retrieval using an index, in: Proceedings of the Fourth International Workshop on Query Processing and Multimedia Issues in Distributed Systems (QPMIDS 2000), IEEE Computer Society, London/Greenwich, UK, 2000, pp. 930–934.
[32] M. Ley, The digital bibliography & library project. URL <http://www.informatik.uni-trier.de/~ley/db/> (valid as of October 3, 2008).

**Ilaria Bartolini** is currently an Assistant Professor with the DEIS department of the University of Bologna (Italy). She graduated in Computer Science (1997) and received a Ph.D. in Electronic and Computer Engineering (2002) from the University of Bologna. In 1998 she spent six months at CWI (Centrum voor Wiskunde en Informatica) in Amsterdam (The Netherlands) as a junior researcher. In 2004 she was a visiting researcher for three months at NJIT (New Jersey Institute of Technology) in Newark, NJ, USA. From January 2008 to April 2008 she was invited visiting professor at the Hong Kong University of Science and Technology (HKUST). Her current research mainly focuses on collaborative filtering, learning of user preferences, similarity and preference query processing in large databases, and retrieval and browsing of image collections. She has published about 30 papers in major international journals (including IEEE TPAMI, ACM TODS, DKE, and MTAP) and conferences (including VLDB, ICDE, PKDD, and CIKM). She served in the program committee of several international conferences and workshops. She is a member of ACM SIGMOD.

**Paolo Ciaccia** has a "Laurea" degree in Electronic Engineering (1985) and a Ph.D. in Electronic and Computer Engineering (1992), both from the University of Bologna, Italy, where he has been Full Professor of Information Systems since 2000. He has published about 100 refereed papers in major international journals (including IEEE TKDE, IEEE TSE, ACM TODS, ACM TOIS, IS, DKE, and Biological Cybernetics) and conferences (including VLDB, ACM-PODS, EDBT, and ICDE). His current research interests include similarity and preference-based query processing, image retrieval, P2P systems, and data streams. He was one of the designers of the *M*-tree, an index for metric data used by many multimedia and data mining research groups in the world. He is a member of IEEE.

**Irene Ntoutsi** received her Diploma (2001) from the Computer Engineering and Informatics Department, University of Patras, Greece. She also holds an M.Sc. (2003) in Computer Science from the same university. In 2008 she received her Ph.D. in Informatics from the Department of Informatics, University of Piraeus, Greece. Her current research interests include data mining, dissimilarity assessment, pattern management and machine learning. She co-authored many papers published in international conferences and journals (like KDD, SDM, PKDD).

**Marco Patella** got the 'Laurea' degree in Electronic Engineering from the University of Bologna, Italy. He received a Ph.D. in Electronic and Computer Engineering (1999) from the same University. Since 2006 he has been Associate Professor at University of Bologna with DEIS. His current research interests include similarity-based query processing in multimedia databases and Data Mining techniques. He is one of the designers of the *M*-tree, an index for metric data, which is used by several multimedia and data mining research groups in the world. He has published more than 25 papers, in the area of database systems, in major international journals (including IEEE TPAMI and ACM TODS) and international conferences (including VLDB, ACM-PODS, EDBT, and ICDE). He has also been part of the program committee of several international conferences and workshops.

**Yannis Theodoridis** is Associate Professor with the Department of Informatics, University of Piraeus (UniPi), where he leads the Information Systems Lab. (http://infolab.cs.unipi.gr). Born in 1967, he received his Diploma (1990) and Ph.D. (1996) in Electrical and Computer Engineering, both from the National Technical University of Athens, Greece. His research interests include spatial and spatiotemporal databases, geographical information management, knowledge discovery and data mining. Currently, he is scientist in charge for UniPi in the EC-funded GeoPKDD project (2005–2008) on geographic privacy-aware knowledge discovery and delivery, also involved in several national-level projects. He has co-authored three monographs and over 50 articles in scientific journals (including Algorithmica, ACM Multimedia and IEEE TKDE) and conferences (including ACM SIGMOD, PODS, VLDB and ICDE) with over 400 citations in his work. He serves on the editorial board of the International Journal on Data Warehousing and Mining. He is member of ACM and IEEE.