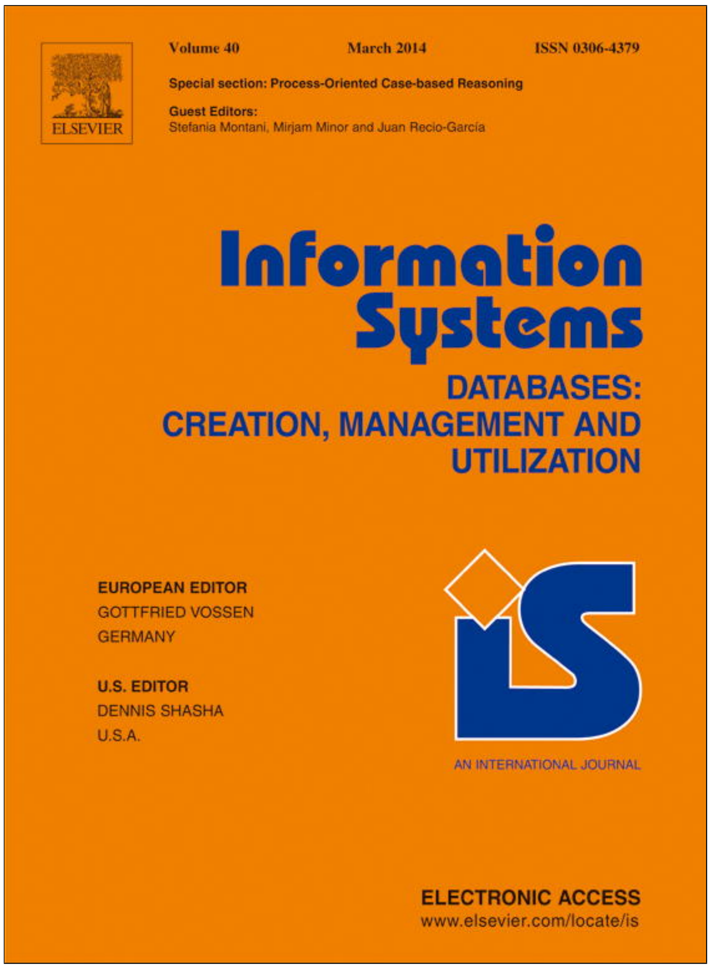

Volume 40 March 2014 ISSN 0306-4379

Special section: Process-Oriented Case-based Reasoning

Guest Editors:
Stefania Montani, Mirjam Minor and Juan Recio-García

Information Systems

DATABASES: CREATION, MANAGEMENT AND UTILIZATION

EUROPEAN EDITOR
GOTTFRIED VOSSEN
GERMANY

U.S. EDITOR
DENNIS SHASHA
U.S.A.

AN INTERNATIONAL JOURNAL

ELECTRONIC ACCESS
www.elsevier.com/locate/is

Information Systems 40 (2014) 128–141

Contents lists available at ScienceDirect

Information Systems

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

Retrieval and clustering for supporting business process adjustment and analysis

Stefania Montani*, Giorgio Leonardi

DISIT, Institute of Computer Science, Università del Piemonte Orientale, Viale Michel 11, I-15121 Alessandria, Italy

ARTICLE INFO

Available online 21 December 2012

Keywords:
Business process adjustment and analysis
Case-based retrieval
Hierarchical clustering
Temporal constraints

ABSTRACT

In this paper, we describe a framework able to support run-time adjustment and a posteriori analysis of business processes, which exploits the retrieval step of the Case-based Reasoning (CBR) methodology. In particular, our framework allows to *retrieve* traces of process execution similar to the current one. Moreover, it supports an automatic organization of the trace database content through the application of hierarchical *clustering* techniques. Results can provide help both to end users, in the process execution phase, and to process engineers, in (formal) process conformance evaluation and long term process schema redesign.

Retrieval and clustering rely on a distance definition able to take into account temporal information in traces. This metric has outperformed simpler distance definitions in our experiments, which were conducted in a real-world application domain.

© 2012 Elsevier Ltd. All rights reserved.

1. Introduction

Business Process (BP) Management is a set of activities aimed at defining, executing and optimizing BP, with the objective of making the business of an enterprise as effective and efficient as possible, and of increasing its economic success. Such activities are highly automated, typically by means of workflow management systems, also called Business Process Management System (BPMS) [1,2].

In normal conditions, a BPMS allows to automatically execute a BP according to its process schema, i.e., to a formalized model in which the actions to be performed and the control flow relations to be respected among them are specified. However, BP optimization may ask the enterprise to be able to flexibly change and adapt such a predefined process schema, in response to expected situations (e.g., new laws, reengineering efforts) as well as to unanticipated exceptions and problems in the operating environment (e.g., emergencies) [3].

The *agile workflow* technology [4,5] is the technical solution which has been invoked to deal with such adaptation and overriding needs. It can support both ad hoc adjustments of individual process instances [6,7], operated by end users, and redesign at the general process schema level, operated by process engineers—applicable even if the default process schema is already in use by some running instances [6,8].

In order to provide an effective and quick adaptation support, many agile workflow systems share the idea of recalling and reusing concrete *examples of changes* adopted in the past. To this end, Case-based Reasoning (CBR) [9] has been proposed as a natural methodological solution. CBR is a reasoning paradigm that exploits the specific knowledge of previously experienced situations, called *cases*. It operates by *retrieving* and *reusing* similar cases in order to solve the problem at hand (after a possible *revision* of the retrieved solutions, if needed). Indeed CBR is particularly well suited for managing exceptional situations, even when they cannot be foreseen or preplanned. As a matter of fact, in the literature

* Corresponding author. Tel.: +39 0131360158; fax: +39 0131360198.
E-mail address: stefania.montani@unipmn.it (S. Montani).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.11.006

cases have often been resorted to in order to describe exceptions, in various domains (see e.g. [10]), and many examples of CBR-based process change support have been proposed (see e.g. [7,11–15]).

The implementation of a proper CBR-based support for process adjustment and analysis obviously starts from a careful evaluation of past cases representation. In many applications, past examples of change are recorded as traces of execution [16] (stored in a database, also known as event log [17]), i.e., as the sequence of the process actions that were actually performed, often coupled with their starting and ending time. In the simplest form, a trace does not log any other feature about the executed actions (e.g., the actor, or the available resources). Moreover, usually it does not provide any contextual information, which could justify the reasons for possible deviations from the prescriptions of the default process schema.

In this paper, we propose a support to BP adjustment and analysis which adopts the retrieval step of the CBR methodology, specifically designed to work on cases in the form of traces of execution.

In our framework, *retrieval* is meant to help *end users* in the process execution phase, when dealing with an atypical situation. Indeed, suggestions on how to adjust the default process schema in the current situation may be obtained by analyzing the most similar retrieved examples of change, recorded as traces that share the starting sequence of actions with the current query (i.e., with the input problem).

Moreover, we support an automatic organization of the case base content (i.e., of all the available traces) through the application of hierarchical *clustering* techniques. Clustering can serve as a starting point for a set of a posteriori trace analyses. In particular, it can help *process engineers* in conformance evaluation (e.g., it can be an input to formal verification of the conformance of traces to proper semantic constraints [18]). Additionally, since changes can also be due to a weak or incomplete initial process schema definition, engineers can exploit (retrieval and) clustering results to draw some suggestions on how to redesign process schemata, in order to incorporate the most frequent and significant changes once and for all.

In our work retrieval and clustering rely on a distance definition able to take into account *temporal information*.

Interestingly, only a few metrics specifically designed to work on traces have been described in the literature. Moreover, most of them do not manage temporal information; in particular, a properly way of comparing qualitative temporal constraints is usually not provided (see Section 5).

On the other hand, neglecting time can be a significant flaw. In fact, time is really crucial in some applications. In medicine, for instance, the role of time is clearly central: it is mandatory to penalize the fact that the very same action had different durations in two traces, or was delayed, especially if referring to emergency procedures. And, generally speaking, temporal information is relevant in all domains, as it can be used e.g., to discover bottlenecks and to measure service levels [17].

The metric we introduce in this work allows us to explicit manage temporal information in traces, paying attention both to quantitative and to qualitative constraints.

In addition to this methodological contribution, in the paper we also describe our experimental work in the field of stroke care, in which we compared the new metric to a classical existing one (namely, the edit distance [19,20]), and to simpler versions of our distance definition, able to manage only part of the overall information available on traces.

The paper is organized as follows. Section 2 presents technical details of the framework. Section 3 describes experimental results. Section 4 adds information about recent methodological improvements we are providing, in order to enhance the framework performance. Section 5 addresses some comparisons with related works. Finally, Section 6 is devoted to conclusions, discussion of limitations and future research directions.

2. A framework for supporting BP adjustment and analysis

This section describes methodological and technical details of our framework.

In particular, central tasks that all CBR tools have to deal with are [9] to define the notion of case, and to find a past case similar to the input one (i.e., to implement retrieval). Case definition and retrieval methods can vary considerably [9], and should be tailored to the needs of the application domain. Specifically, a proper distance definition has to be introduced, in order to optimize the reliability of retrieval results. The quality of clustering output also strongly depends on the distance we define.

In our work, the notion of case is related to the one of trace [16]. As for the distance definition, as observed in the Introduction, our goal has been the one to explicitly manage temporal information too, thus overcoming the limitations of many existing works.

Our notion of case and our case distance definition are illustrated in Section 2.1. Section 2.2 then moves to the choice of specific retrieval and clustering approaches.

2.1. Case representation and distance definition

We define a *case* as a trace of execution of a given process schema. In particular, every trace is a sequence of actions, each one stored with its execution starting and ending times. Additional action features (e.g., actors, available resources) are not recorded in our framework. Therefore, an action is basically just a symbol (plus the temporal information).

The only type of control flow structure we explicitly deal with is *sequence*, since traces record what has already been executed. This means that alternatives are not possible, and iterations are completely unfolded. Partial or complete parallelism between actions may have taken place, and can be derived from action starting and ending times. Indeed, starting and ending times allow to get information about action durations, as well as qualitative (e.g., Allen's *before*, *overlaps*, *equals*, etc. [21]) and quantitative temporal constraints (e.g., delay length, overlap length [22]) between pairs of consecutive actions.

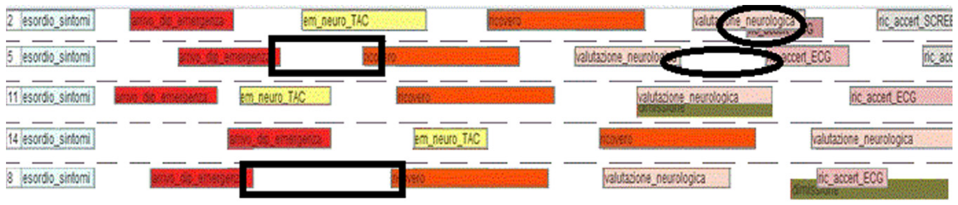

Fig. 1. A view of (part of) five real traces, referring to the domain of stroke management. Identical actions are depicted in the same color (e.g., yellow actions are Computed Assisted Tomography), but can have a different duration (e.g., Computed Assisted Tomography lasts longer in the first trace). Examples of different qualitative constraints are provided as well. For instance, pink and purple actions overlap in trace 1, while pink action is before purple action in trace 2 (see round shapes). On the other hand, red action is before orange action both in trace 2 and in trace 5, but the length of the delay in between them is different in the two cases (see square shapes). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

For example, two completely parallel actions are completely overlapping actions (i.e., Allen's *equals* [21]).

Fig. 1 provides a view of (part of) five real traces, referring to the domain of stroke management. Identical actions are depicted in the same color (e.g., yellow actions are Computed Assisted Tomography), but can have a different duration (e.g., Computed Assisted Tomography lasts longer in the first trace). Examples of different qualitative constraints are provided as well.

If we want to take into account all the types of information we can find on traces, we need to calculate *distance* on the basis of both:

- atemporal information (i.e., action types);
- temporal information (i.e., action durations, qualitative and quantitative constraints between pairs of consecutive actions).

Indeed, temporal information is generally very relevant, because it can be used e.g., to discover bottlenecks and to measure service levels [17]. Moreover, in many domains (in particular medical ones, in which the role of time is often central) it is mandatory to penalize the fact that the very same action had different durations in two traces, or was delayed (in fact, anomalous action lengths and delays have to be justified, e.g., for legal purposes).

Operatively, we first take into account action types, by calculating a modified edit distance which we have called *Trace Edit Distance*. As the classical edit distance [19], our metric tests all possible combinations of editing operations that could transform one trace into the other one (see Section 2.1.1). Then, it takes the combination associated to the minimal cost. Such a choice corresponds to a specific alignment of the two traces, in which each action in one trace has been matched to an action in the other trace—or to a gap. We will call it the *optimal alignment* henceforth.

Given the optimal alignment, we can then take into account temporal information. In particular, we compare the durations of aligned actions by means of a metric we called *Interval Distance*.

Moreover, we take into account the temporal constraints between two pairs of subsequent aligned actions on the traces being compared (e.g., actions A and B in trace P; the aligned actions A' and B' in trace Q). We quantify the distance between their qualitative constraints (e.g., A and B overlap in trace P; A' meets B' in trace Q), by resorting to a metric known as *Neighbors-graph Distance*. If Neighbors-graph Distance is 0, because the two pairs of actions share the same qualitative constraint (e.g., A and B overlap in trace P; A' and B' also overlap in trace Q), we compare quantitative constraints by properly applying Interval Distance again (e.g., by calculating Interval Distance between the two overlap lengths).

These three contributions (i.e., Trace Edit Distance, Interval Distance between durations, Neighbors-graph Distance or Interval Distance between pairs of actions) are finally combined as a linear combination with non-negative weights (by now, we chose identical weights for all contributions).

Formal definitions of Trace Edit Distance, Interval Distance and Neighbors-graph Distance are provided below.

2.1.1. *Trace Edit Distance*

In order to calculate Trace Edit Distance, we only consider non-temporal information, i.e., we work on traces as if they simply were strings of symbols, with every action corresponding to a symbol.

We define a set of edit operations on traces. Each edit operation performs a modification of the following kinds: (i) *substitute* one action with a different one, (ii) *insert* a new action, or (iii) *delete* an action.

However, in our approach, the cost of a *substitution* is not always set to 1, as in the classical edit distance [19]. In fact, as in the weighted edit distance (see e.g. [23]), we define it as a value $\in [0,1]$ which depends on what action appears in a trace as a substitution of the corresponding action in the other trace. In particular, we organize actions in a *taxonomy*, on the basis of domain knowledge. The closer two actions are in the taxonomy, the less penalty has to be introduced for substitution ([24]; see also [25–27]).

In detail, in our work substitution penalty is set to the *Taxonomic Distance* between the two actions [24], i.e., to the normalized number of arcs on the path between the two actions in the taxonomy:

Definition 1 (*Taxonomic Distance*). Let α and β be two actions in the taxonomy t, and let γ be the closest common ancestor of α and β. The *Taxonomic Distance*

$dt(\alpha,\beta)$ between α and β is defined as

$$dt(\alpha,\beta) = \frac{N_1+N_2}{N_1+N_2+2*N_3}$$

where N_1 is the number of arcs in the path from α and γ in t, N_2 is the number of arcs in the path from β and γ, and N_3 is the number of arcs in the path from the taxonomy root and γ.

The Trace Edit Distance $\text{trace}_{NGLD}(P,Q)$ is finally calculated as the Normalized Generalized Levenshtein Distance (NGLD) [20] between two traces P and Q (interpreted as two strings of symbols). Formally, we provide the following definitions:

Definition 2 (*Trace Generalized Levenshtein Distance*). Let P and Q be two traces of actions, and let α and β be two actions. The Trace Generalized Levenshtein Distance $\text{trace}_{GLD}(P,Q)$ between P and Q is defined as

$$\text{trace}_{GLD}(P,Q) = \min\left\{\sum_{i=1}^{k} c(e_i)\right\}$$

where $(e_1,\dots,e_k)$ transforms P into Q, and

- $c(e_i)=1$, if e_i is an action insertion or deletion;
- $c(e_i)=dt(\alpha,\beta)$, if e_i is the substitution of α (appearing in P) with β (appearing in Q), with $dt(\alpha,\beta)$ defined as in Definition 1 above.

As already observed, the minimization of the sum of the editing costs allows to find the optimal alignment between the two traces being compared.

Definition 3 (*Trace Edit Distance (Trace Normalized Generalized Levenshtein Distance)*). Let P and Q be two traces of actions, and let $\text{trace}_{GLD}(P,Q)$ be defined as in Definition 2 above. We define Trace Edit Distance $\text{trace}_{NGLD}(P,Q)$ between P and Q as

$$\text{trace}_{NGLD}(P,Q) = \frac{2*\text{trace}_{GLD}(P,Q)}{|P|+|Q|+\text{trace}_{GLD}(P,Q)}$$

where $|P|$ and $|Q|$ are the lengths (i.e., the number of actions) of P and Q, respectively.

$\text{trace}_{NGLD}(P,Q)$ is just an application of NGLD [20] to traces. Interestingly, it has been proved [20] that NGLD is a metric. In particular, unlike other definitions of normalized edit distance (e.g. [28]), it also preserves the triangle inequality.

Like every variant of the classical edit distance described in the literature, $\text{trace}_{NGLD}(P,Q)$ can be calculated resorting to a dynamic programming approach, making its complexity tractable [20] (see however our work on retrieval time improvement in Section 4).

2.1.2. Interval Distance

In our approach, action duration is represented as the length of the interval bounded by the starting and ending point of the action itself; starting and ending points of two consecutive actions also allow to identify the type of qualitative constraint between the actions themselves (e.g., Allen's interval relations [21] *meets*, *before*, *overlaps*, etc.), and to quantify it (e.g., length of the delay, extension of the overlap). Delay lengths and overlap extensions are interval lengths as well.

In order to compare the lengths of matching intervals we resort to a metric we have called Interval Distance, defined as follows:

Definition 4 (*Interval Distance*). Let i and j be two intervals of length len_i and len_j, respectively, and let maxlen be the length of the longest interval available in our trace database. The Interval Distance $\text{interval}_d(i,j)$ between i and j is defined as

$$\text{interval}_d(i,j) = \frac{|\text{len}_i - \text{len}_j|}{\text{maxlen}}$$

It is worth stressing that Interval Distance is applied to compare the durations of two aligned actions (according to the optimal alignment—e.g., the yellow actions in traces 1 and 3, see Fig. 1). Moreover, it is also exploited to compare the lengths of intervals in between aligned actions in the traces (see Fig. 1—square shapes), or the lengths of two corresponding overlaps. In the case of delay comparisons, maxlen is set to the length of the longest delay logged in our trace database.

Once Interval Distance has been calculated referring to all the actions in the two traces being compared, the obtained contributions are summed up. Finally, we divide by the length of the longest trace in the database (in terms of number of actions).

Given such a definition, it is straightforward to prove that Interval Distance is a metric.

2.1.3. Neighbors-graph Distance

When comparing two pairs of corresponding actions, it may happen that they do not share the same qualitative constraint (see Fig. 1—round shapes). In this case, we cannot resort to Interval Distance to compare the interaction constraints, because they have a different semantic meaning. On the other end, we can quantify the difference between the two qualitative constraints by resorting to the *A-neighbors graph* proposed by Freska [29] (see Fig. 2).

On such a graph, we can define the Neighbors-graph Distance, as follows:

Definition 5 (*Neighbors-graph Distance*). Let i and j be two Allen's temporal relations [21], and let G be the A-neighbors graph in Fig. 2. The *Neighbors-graph Distance* $\text{ngraph}_d(i,j)$ between i and j is defined as

$$\text{ngraph}_d(i,j) = \frac{\text{path}(i,j,G)}{\max_{k,l\in G}\{(\text{path}(k,l,G))\}}$$

where $\text{path}(i,j,G)$ measures the shortest path on G between i and j, and $\max_{k,l\in G}\{(\text{path}(k,l,G))\}$ normalizes the distance considering the longest path on G.

As above, once Neighbors-graph Distance has been calculated referring to all the actions in the two traces being compared, the obtained contributions are summed up. Finally, we divide by the length of the longest trace in the database (in terms of number of actions).

Given such a definition, it is straightforward to prove that Neighbors-graph Distance is a metric.

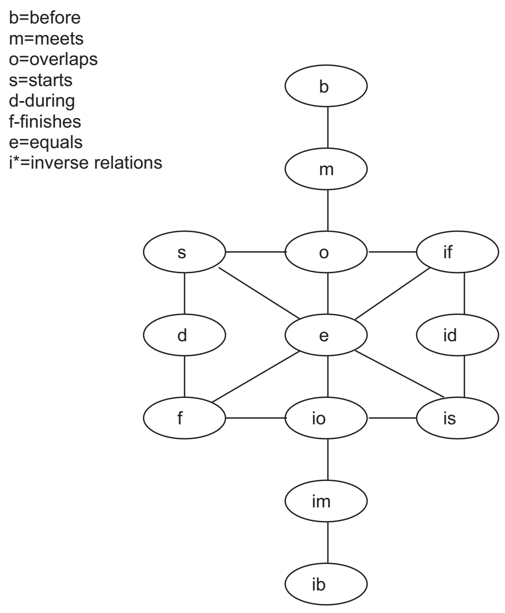

Fig. 2. The *A-neighbors graph* proposed by Freska [29]. The more two qualitative constraints are similar, the shorter is the path between them on the graph.

The final weighted average of Trace Edit Distance, Interval Distance and Neighbors-graph Distance is then a metric as well.

2.2. Retrieval and clustering techniques

Given the distance definition illustrated above, our framework exploits it to implement classical methodologies for retrieval and clustering.

In particular, trace retrieval is performed by a K-Nearest Neighbor technique, consisting in identifying the closest *k* cases (i.e., traces) with respect to an input one, according to the distance definition we have introduced. Clearly, the value of *k* is a critical parameter, which has to be (experimentally) set according to the specific application domain needs.

The clustering facility we have implemented resorts to a hierarchical clustering technique, known as Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [30]. UPGMA is typically applied in bioinformatics, where sequences of symbols (similar to our traces) have to be compared. The algorithm operates in a bottom-up fashion. At each step, the nearest two clusters are combined into a higher-level cluster. The distance between any two clusters A and B is taken to be the average of all distances between pairs of objects “x” in A and “y” in B, that is, the mean distance between elements of each cluster. After the creation of a new cluster, UPGMA properly updates a pairwise distance matrix it maintains. UPGMA also allows to build the phylogenetic tree of the obtained clusters, which can be resorted to for user-friendly visualization purposes, very useful in our domain.

3. Experimental results

In this section we will provide some results on clustering experiments. Some experiment on retrieval will be presented in Section 4, where we will discuss our most recent improvements.

All of our experiments were conducted working on real patient traces taken from the stroke management domain. Actually, Health-Care Organizations (HCO) place strong emphasis on efficiency and effectiveness, to control their health-care performance and expenditures: they thus need to evaluate existing infrastructures and the services provided. To perform this evaluation, it is crucial to explore the data collected by the HCO systems, organizing them in the form of traces, which can be seen as the history of what happened in the HCO itself. Traces can be helpful to gain a clear picture of the actual care process, through the use of techniques like the ones introduced in this paper. These considerations motivated the choice of a medical application as an example of business process.

In particular, in our experiments we aimed at verifying whether the distance function described in this paper was able to overcome the performance of classical edit distance, and of simpler versions of our distance definition. Namely, we have considered Trace Edit Distance alone, which manages only non-temporal information, as well as a metric that combines Trace Edit Distance and Interval Distance. This last metric can manage action durations and delays between actions, but is unable to treat qualitative constraints other than *before* and *meets*, and is unable to make comparisons between different qualitative constraints.

The hypothesis we wished to test was the following: including domain knowledge (through Trace Edit Distance) allows to obtain more homogeneous and compact clusters (i.e., able to aggregate closer examples); including temporal information provides even a further homogeneity improvement (moreover, obviously, the use of Neighbors-graph Distance allows to deal with all kinds of traces, and not only with strictly sequential ones).

The database on which we made our experiments was composed of 100 traces collected at one of the largest stroke management units in the Lombardia region, Italy.

Details of the application domain and experimental results are provided below.

3.1. Stroke management

A stroke is the rapidly developing loss of brain function(s) due to disturbance in the blood supply to the brain. This can be due to ischemia (lack of glucose and oxygen supply) caused by thrombosis or embolism, or to a hemorrhage. As a result, the affected area of the brain is unable to function, leading to inability to move one or more limbs on one side of the body, inability to understand or formulate speech, or inability to see one side of the visual field. A stroke is a medical emergency and can cause permanent neurological damage, complications, and death. It is the leading cause of adult disability in the United States and Europe. It is the number two cause of death worldwide and may soon become the leading one.

The best medical practice [31] requires that stroke patients are treated according to a management protocol, which is basically composed by four steps:

1. emergency management;
2. hospitalization;
3. dismissal;
4. follow up.

Each step is in turn composed by a sequence of actions, which must respect some criteria, although inter-patients and inter-hospitals variations are admissible.

In particular, in step 1, symptoms onset must be recognized, the patient must be taken to the hospital, and a brain Computer Assisted Tomography must be executed. In step 2, diagnosis has to be finalized, by means of a neurological evaluation and of several additional diagnostic investigations, meant to confirm the stroke hypothesis. Diagnostic procedures may vary, but most patients undergo electrocardiogram and chest X-ray. Finally, a proper therapy has to be initiated: up to 90% patients are treated with antiaggregants. Rehabilitation also must be started as soon as possible during hospitalization.

In our experiments, we used traces collected on real patients, detailing the actions of steps 1 and 2 (see also Fig. 1 for some examples).

3.2. *Comparing different distance measures*

We compared the clustering results obtained by adopting four different distance measures, namely

- (1) normalized edit distance [19,20];
- (2) Trace Edit Distance (see Definition 3 in Section 2.1);
- (3) a distance measure that linearly combines Trace Edit Distance (see Definition 3 in Section 2.1) and Interval Distance (see Definition 4 in Section 2.1.2). Such a distance manages durations and delays between actions, but is unable to treat qualitative constraints other than *before* and *meets*, and is unable to compare different qualitative constraints;
- (4) the more complete distance measure introduced in Section 2.1.

We made our experiments working on two databases:

- DB1, in which some traces were modified, in order to remove total or partial overlaps between actions; this change was obtained by reducing the duration of some actions;
- DB2, containing the original, real-world traces collected in Lombardia.

The creation of DB1 as a modification of DB2 was needed because distances (1), (2) and (3) cannot deal with traces with overlapping actions (only distance (4) is general enough to treat this issue).

Both databases contained 100 traces.

3.2.1. *Quantitative results*

Fig. 3 shows part of the cluster hierarchies we obtained by applying distances (1) and (2), respectively, on DB1 (identical results were obtained on DB2, since temporal information is ignored by these distances).

We can observe that the structure of the hierarchies and the content of the resulting clusters is very different in the two situations.

In particular, the hierarchy built using classical edit distance (distance (1)—see Fig. 3, upper part) is very unbalanced: every node is split into two children, one of which usually corresponds to a very big cluster (containing most of the traces of its parent node), while the other contains just a few traces.

On the other hand, the hierarchy built resorting to Trace Edit Distance (distance (2)—see Fig. 3, lower part)

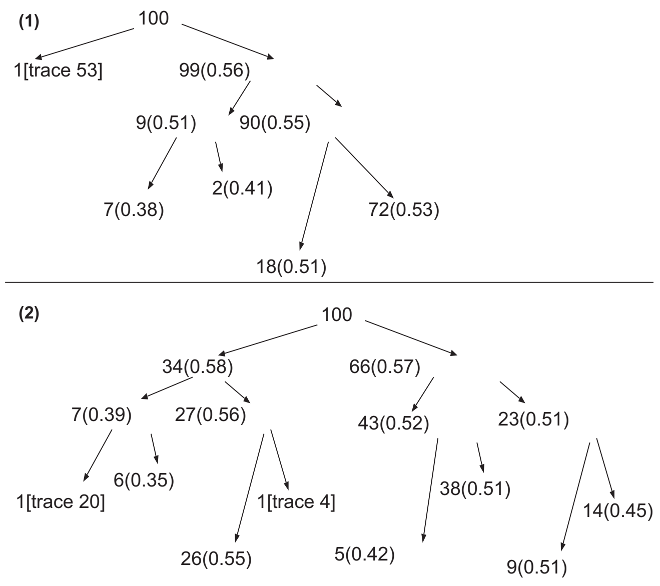

Fig. 3. Part of the cluster hierarchies obtained by applying distance (1) (top) and distance (2) (bottom) on DB1. Every node represents a cluster and reports the number of traces in the cluster itself, and their average normalized edit distance (in brackets).

appears to be much more balanced, and every node is normally split into two clusters of more comparable dimensions. This is probably due to the strong penalty assigned by distance (1) to all substitutions; such a penalty often isolates some traces as anomalous ones (see also the discussion on trace no. 53 below).

We also studied the cluster contents, in order to verify cluster *homogeneity*. Homogeneity is a widely used measure of the quality of the output of a clustering method (see e.g. [32–35]). A classical definition of cluster homogeneity is the following [32]:

$$H(C) = \frac{\sum_{x,y \in C}(1 - dist(x,y))}{\binom{|C|}{2}}$$

where $|C|$ is the number of elements in cluster C, and $1-\text{dist}(x,y)$ is the similarity between any two elements x and y in C. Note that, in the case of singleton clusters, homogeneity is set to 1 (see e.g. [35]).

The higher the homogeneity value, the better the quality of clustering results.

The average of the homogeneity H of the individual clusters can be calculated on (some of) the clusters obtained through the method at hand, in order to assess its quality. Average cluster homogeneity allows to compare the output of different clustering techniques on the same dataset (or the output obtained by differently setting up the same clustering technique, as we did by running UPGMA with different distance measures).

An appropriate definition of $dist(x,y)$ is problem dependent [32]. In our domain, we exploited the normalized edit distance between pairs of traces. The choice of the edit distance allowed us to compare our more complex distance measures with a very classical metric, used as a common reference.

We calculated the homogeneity of the clusters obtained using distances (1) and (2). We then computed the average of cluster homogeneity values level by level in the two hierarchies.

Finally, we calculated an additional indicator, namely the average number of traces inside a cluster not exceeding a normalized edit distance of 0.5.[1]

Results are reported in the first two columns of Table 1.

Specifically, clusters obtained using distance (1) had an average homogeneity of 0.47 at level 3 of the hierarchy, while using distance (2) they reached an average homogeneity of 0.50. A similar trend was obtained if working at other intermediate levels of the hierarchy (not reported due to space constraints).

Additionally, with distance (2), the percentage of pairs of traces with a distance <0.5 was 37% on average at level 3 of the hierarchy, while it dropped to 28% using distance (1).

[1] This less standard indicator was calculated to reinforce the homogeneity results; note that the choice of 0.5 could be substituted by a different one.

Table 1
Average homogeneity (row 1) and average number of pairs of traces with a distance <0.5 (row 2). All values are referred to clusters at level 3 in the hierarchies.

Distance	(1)	(2)	(3)	(4)	(4-DB2)
Homogeneity	0.47	0.50	0.60	0.62	0.62
Pairs <0.5 (%)	28	37	43	49	52

Fig. 4, on the other hand, shows part of the cluster hierarchies we obtained by applying distances (3) and (4), respectively, on DB1.

As it can be observed in the figure, the hierarchies obtained by applying distances (3) and (4) are very similar (in topology and number of traces contained in the clusters) to the one obtained by applying distance (2). However, they lead to higher homogeneity values. At level 3, homogeneity grows to 0.60 when applying distance (3), and to 0.62 when applying distance (4). Once again, a similar trend was obtained if working at other intermediate levels of the hierarchy. The percentage of pairs of traces with a distance <0.5 also grows progressively (see Table 1).

Finally, Fig. 5 reports part of the hierarchy of clusters obtained by applying distance (4) to the original traces (DB2), which included all kinds of qualitative temporal constraints. Distance (3) cannot be applied to this database.

With distance (4) homogeneity reaches the average value of 0.62 at level 3 on DB2 as well, with an average percentage of pairs of traces with a distance <0.5 of 52% (see Table 1), thus leading to the best overall experimental results.

3.2.2. *Analysis of specific traces*

It is also interesting to comment on specific traces, early isolated in (some of) the cluster hierarchies. In particular, all distances, except distance (2), are able to quickly isolate trace no. 53. Such a trace contains some quite unusual actions as substitutions of more common ones. Specifically, anticoagulant drugs were provided instead of antiaggregant drugs, due to an allergy to acetylsalicylic acid. Distance (1) penalizes this difference, and early isolates the trace; on the other hand, distance (2) does not, because the two actions are very close in the stroke domain taxonomy (indeed, they both describe therapeutic actions with very similar pharmacological effects). The behavior of distance (2) is thus more correct as for the medical semantic meaning, but neither distance (1) nor distance (2) take into account the role of time. And indeed time is relevant in this example, because trace no. 53 is anomalous in its temporal component too (its delays between consecutive actions are often longer than the ones of the other database traces; in fact, the patient was quickly improving, thus justifying a longer observation time between further drug provisions/support therapies). Actually, distance (3) and distance (4), which introduce the temporal contribution, are able to identify such a trace as an anomalous one, and to isolate it quite early in their hierarchies (even if they resort to Trace Edit Distance, i.e.,

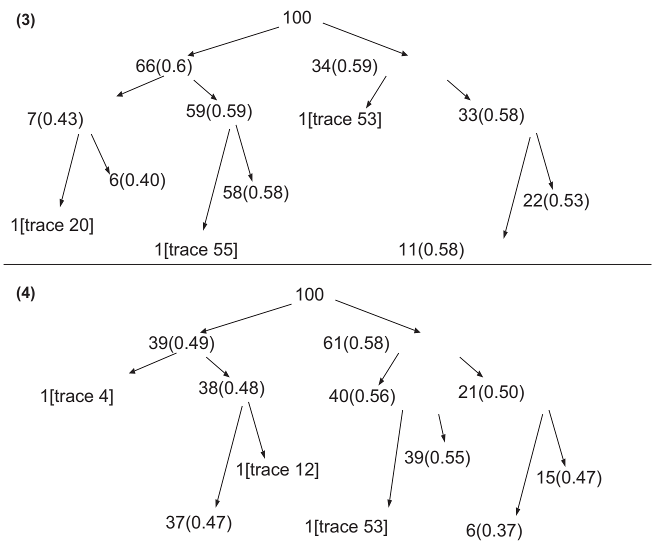

Fig. 4. Part of the cluster hierarchies obtained by applying distance (3) (top) and distance (4) (bottom) on DB1.

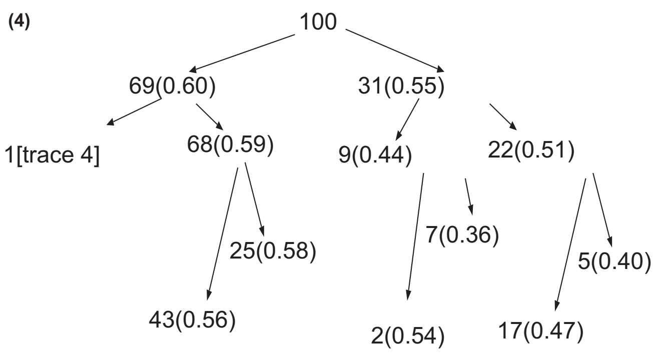

Fig. 5. Part of the cluster hierarchy obtained by applying distance (4) on DB2.

to distance (2), for the atemporal component calculation). Moving to more complete distance definitions thus allows to correctly take into account all the different features (i.e., action types and temporal information) recorded in traces.

Another interesting example, which allows to comment on the difference between distance (3) and distance (4), is represented by trace no. 20. Distance (3) isolates it early in the hierarchy, while distance (4) does not. The reason can be found in the types of qualitative constraints recorded in trace no. 20, which includes many *meeting* consecutive actions. On the other hand, in the majority of the other database traces, one action is typically *before* the next one. Distance (3) does not differentiate between the two types of constraints, it only applies Interval Distance to the delays between consecutive actions (delays that are typically equal to zero in trace no. 20, and longer in other traces). The different contributions obtained on delays allow to identify trace no. 20 as an anomalous one. On the other hand, trace no. 20 does not appear as a peculiar one according to distance (4). In fact, the two qualitative constraints *meets* and *before* are now compared according to Neighbors-graph Distance. Neighbors-graph Distance is low, because the two relations are close in the A-neighbors graph; this contribution, therefore, does not penalize trace no. 20 as much as Interval Distance did.

3.2.3. *Concluding remarks on experiments*

In conclusion, our experiments show that the use of domain knowledge and of temporal information in the distance definition allows to obtain more homogeneous and compact clusters (i.e., able to aggregate closer examples) in the intermediate levels of the hierarchy, which is a desirable results—and a meaningful outcome, in a domain like the one of emergency medicine, in which the role of time is obviously central. As an interesting by-product of clustering, anomalous traces are also correctly isolated (according to the opinion of the domain experts working with us).

In particular, the best homogeneity values are obtained by resorting to distance (4), which also allows to properly compare different qualitative temporal constraints. Of course, distance (4) is also the only one which can be applied to every database, including the one in which (partially) overlapping traces are recorded.

4. Improving performance through a pivoting-based technique

As observed in Section 2, distance calculation is tractable, and indeed retrieval was fast in the experiments we have conducted so far (see [36], where, however, we only relied on Trace Edit Distance, and did not manage

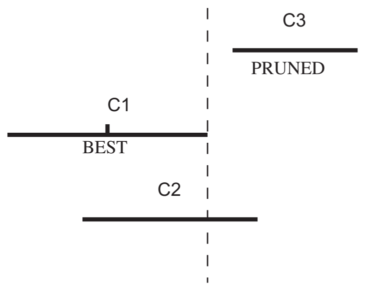

Fig. 6. Bound pruning in PBR.

Table 2
Average query answering time (in milliseconds) for retrieving the best 20 cases, with PBR (column 2) and without PBR (column 3), on 14 case bases of growing dimensions. The average number of pruned cases when resorting to PBR is reported in column 4.

DB dimension	Time with PBR	Time no PBR	Pruned
250	73.02	107.80	96
500	137.28	204.64	247
750	224.48	330.09	359
1000	289.56	419.00	501
1250	373.10	579.50	648
1500	450.97	654.88	860
1750	538.31	773.86	961
2000	585.09	854.68	1107
2250	662.96	979.12	1324
2500	717.49	1076.55	1448
2750	775.03	1158.50	1651
3000	815.46	1269.42	1846
3250	927.17	1413.68	1989
3500	1057.41	1563.17	2083

temporal information); nonetheless, it can become computationally expensive when working on very large databases. This problem has already been highlighted in process retrieval [37].

To this end, we are currently designing and implementing a methodology able to enhance the performance of our tool, avoiding exhaustive search of similar traces. Specifically, we are resorting to *Pivoting-Based Retrieval* (PBR—see e.g. [38,39]), which allows one to focus on particularly promising regions of the search space, and to neglect the others.

The main idea in PBR consists in:

- computing the distance between a representative case (Pivot) and all the other cases (off-line);
- computing the distance between the Pivot and the input case;
- estimating the distance between the input case and all the remaining cases using triangle inequality, thus finding a lower and an upper bound for the distance value.

The intervals whose lower bound is higher than the minimum of all the upper bounds can be pruned (see Fig. 6).

We initialize $BESTp=\infty$ and $SOL=\{\}$. We then apply the following iterative procedure:

```
1. Choose the Pivot case as the minimum of the midpoints
of the intervals; compute the distance between the input
case and the Pivot (DIST); set BEST=DIST;
2. If BESTp > BEST set SOL=PIVOT and BESTp=BEST;
3. Else if BESTp=BEST set SOL={PIVOT, SOL};
4. Prune the intervals whose lower bound is bigger than
BEST, and remove the Pivot from the set of cases
Back to step 1.
```

We have made some first tests by defining the Pivot as the mean case, i.e., the one whose average dissimilarity to all the objects in the database is minimal. Other choices based on heuristics can be considered as well.

Table 2 reports on our experiments on the use of PBR to speed up retrieval time. We made tests on different case base dimensions (from 250 to 3500 traces; artificial traces were generated randomly to this end). On every case base, we executed 400 queries. Table 2 compares the average query answering time for retrieving the best 20 cases, with PBR (column 2) and without PBR (column 3). The average number of pruned cases when resorting to PBR is reported as well (column 4). Experiments were performed on an Intel Core 2 Duo T9400, equipped with 4 GB of DDR2 RAM. Times are in milliseconds. As it can be observed, up to 60% of the original traces could be pruned in some situations, and retrieval time always strongly improved (see also Fig. 7).

We also plan to implement a more complex solution, in which we will first cluster the available traces (e.g., resorting to the well-known K-Means algorithm [40]), and then select one Pivot for each cluster (i.e., the cluster mean). Specifically, we will perform a multi-step retrieval, in which:

- we identify the cluster the input case should be assigned to;
- we apply the PBR procedure described above to the cluster at hand (taking its mean as the initial Pivot).

An extensive experimental work will then follow, in order to test the advantages of this enhanced PBR procedure with respect to the standard PBR technique described above, and to exhaustive search. Moreover, we plan to carefully evaluate the trade-off between the computational advantages of clustering-based early pruning, and the risk of loosing close neighbors of the input case, which belong to different clusters.

As a future research direction, we also plan to work on how to apply a retrieval method based on cover trees [41] to our domain. Indeed, such a method can potentially lead to further significant computational improvements.

5. Related works

Examples of CBR tools in BP management, and specifically in process adjustment support, are described in the literature (e.g. [11,12,7,14,15]); a few works exploiting clustering techniques are reported as well (e.g. [42,43])—even though they mainly deal with process mining [44] (see below).

Since the main methodological contribution of our work consists in the definition of a proper distance function, in our comparison with the existing literature we will first focus on this issue, and on the papers providing interesting solutions in relation to it. We will

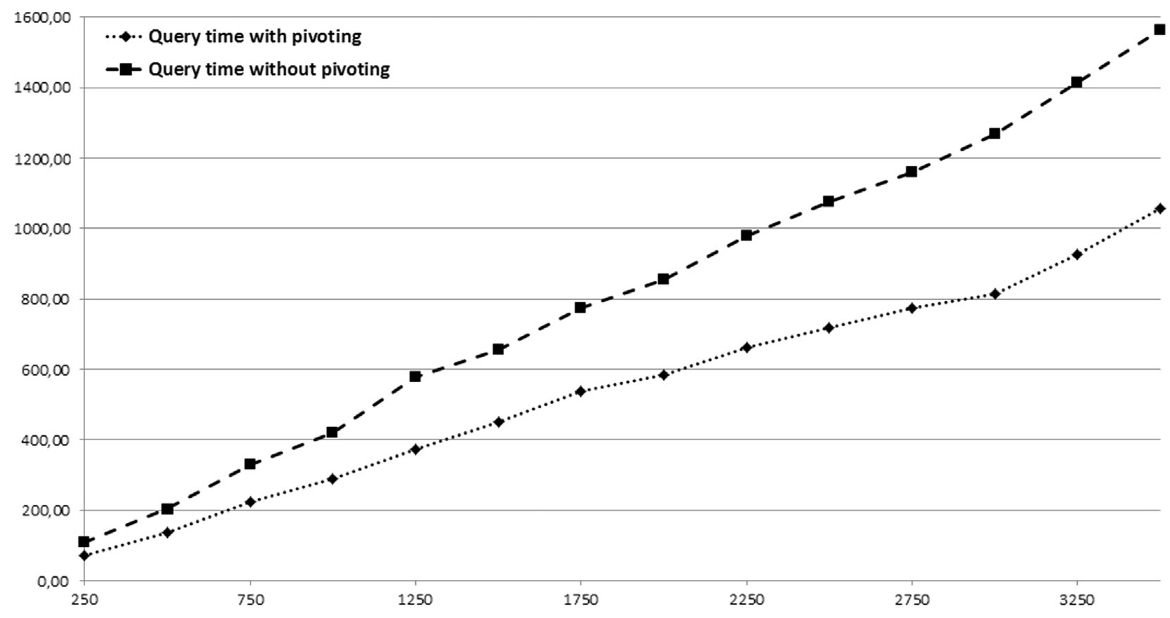

Fig. 7. Comparison between the average query answering time for retrieving the best 20 cases, with PBR and without PBR, on 14 case bases of growing dimensions—see Table 2 for numerical details.

then move to an analysis of works in process mining, in order to clarify the possible connections and the borders between our contribution and this research area. Finally, we will discuss some contributions on BP management in health care applications.

5.1. *Distance functions*

A number of distance measure definitions for agile workflows exist. However, these definitions typically require further information in addition to the workflow structure, such as semantic annotations [45], or conversational knowledge [4,7]. Such approaches are usually context-aware, that is, the contextual information is considered as a part of the similarity assessment of workflows. Unfortunately, any contextual information, as well as conversational knowledge, is not always available, especially when instances of process execution are recorded as traces of actions. Starting from this observation, a rather simple graph edit distance measure [46] has been proposed and adapted for similarity assessment in workflow change reuse [14].

Our approach somehow moves from the same graph edit distance definition. However, with respect to the work in [14], by focusing just on traces of execution we do not need to deal with control flow elements (such as alternatives and iterations). As a matter of fact, traces are always linear, i.e., they just admit the sequence control flow element. From this point of view, our approach is thus simpler than the one in [14].

On the other hand, when focusing on linear traces our approach is more general and flexible. Indeed, we resort to taxonomic knowledge for comparing pairs of actions, so that two different actions do not always have a zero similarity. Moreover, we have introduced a distance definition which also allows to take into account qualitative and quantitative temporal constraints between actions in process logs. Such a capability is not provided at all in [14].

On the other hand, a treatment of temporal information in trace distance calculation has been proposed in [47]. Somehow similarly to our approach, the distance defined in that work combines a contribution related to action similarity, and a contribution related to delays between actions. As regards the temporal component, in particular, it relies on an Interval Distance definition which is quite similar to ours. Differently from what we do, however, the work in [47] always starts the comparison from the last two action in the traces: no search for the optimal action alignment is performed. Moreover, it stops the calculation if the distance between two actions/intervals exceeds a given threshold, while we always calculate the overall distance: as a matter of fact, even high distance values are resorted to by our clustering algorithm. The distance function in [47] does not exploit action duration, and does not rely on taxonomical information about actions, as we do. Finally, it does not deal with different types of qualitative temporal constraints, since it cannot manage (partially) overlapping actions. We thus believe that our approach is potentially more flexible in practice.

Another contribution [48] addresses the problem of defining a similarity measure able to treat temporal information, and is specifically designed for clinical workflow traces. Interestingly, the authors consider qualitative temporal constraints between matched pairs of actions, resorting to the A-neighbors graph proposed by Freska [29], as we do. However, in [48] the alignment problem is strongly simplified, as they only match actions with the same name. Our approach thus extends their work.

5.2. *Process mining*

As it is well known, process mining [44] describes a family of a posteriori analysis techniques exploiting the information recorded in traces. As well stated in [17], process mining not only consists in discovering a process schema when it is not available, but also supports conformance analysis and process enhancement. To support conformance analysis, an already known process schema can be compared with a model mined from the available traces, typically by formal conformance checking techniques (see e.g. [49]). Possible deviations can be identified, and be used to support enhancement (e.g., process schema repair or extension). The work in [50], for instance, proposes to use association rules to properly group deviating traces, making the analysis of deviations less complex.

Indeed, our work on clustering could be seen as an alternative approach (with respect to e.g. [50]) to provide an input to the analysis of deviations, in which it is not required to mine a model from traces (as in e.g. [49]). Clusters may group traces conforming to the default schema, but may also highlight the existence of frequent, similar changes in the database. Clustering is a rather simple data mining technique, and does not lead to the extraction of a whole process model; nonetheless, clearly grouping trace changes can be a significant support for e.g. (formal) checking of conformance and verification of semantic properties (but also for visual inspection by human experts).

Clustering and trace alignment can also improve the quality of the process mining activity itself (see e.g. [42,43,51–53]): indeed, by first clustering traces, and then applying process mining within single clusters, it is possible to obtain simpler and less confusing process models. In the available literature approaches meant to provide this functionality, however, temporal information is typically not considered. Our work is therefore more complete.

Our framework could also be taken into account when one has to deal with the concept drift issue in process mining [17]. This issue refers to the fact that, when mining a model from traces, it is normally assumed that the process schema is in a steady state; however, processes often change over time, and traces embed these changes. Classical process mining techniques do not take into account concept drift, and could provide low quality mining results. A few exceptions are represented by the very recent works in [54,55]. In [54], for instance, the authors define proper features in traces and in the overall event log, and statistically test the changes in feature values to discover concept drift. They mainly focus on changes in ordering relations between actions in traces, while [55] is more interested in added/removed actions. Clustering can be seen as a lazier support to concept drift identification, where traces containing changes are automatically separated from the cluster of traces fully compliant with the default process schema. An analysis of trace temporal information in such clusters could support an identification of concept drifts, and in particular of sudden and recurrent drifts (see [54]). Indeed, [55] suggests the use of clustering techniques to identify concept drift as well, however temporal constraints in traces are not managed in that work (traces are only chronologically ordered). We plan to further investigate this aspect in our future work.

5.3. BP management in health care applications

Interestingly, BP management and process mining techniques have gained particular attention in health care applications in the latest years. Just to cite very recent works, in [56] the complexities of health care processes (i.e., clinical guidelines and pathways), which are human centric, and multi-disciplinary in nature, are well analyzed. In [57] a system able to support adaptations of running health care process instances is described. A lot of attention is also devoted to conformance checking techniques of clinical guidelines and pathways (see e.g. [58,59]). The works in [60,52,61,53] deal with process mining in health care. Specifically, [60,52] apply process mining techniques to the stroke management and to the gynecological oncology domains, respectively. The contributions in [52,61,53] also propose pre-processing techniques to improve process mining (which are highly recommended in the medical domain, due to its complexity, as discussed in the already cited [56]).

In particular [53] proposes a clustering approach to pre-process traces, where each cluster is based on a probabilistic model, namely a first-order Markov chain. Such a work also focuses on the identification of anomalous traces, that can be found as belonging to clusters with a very low support. Indeed, anomalous traces can be highlighted by hierarchical clustering techniques as well (see Section 3). A more systematic analysis of such examples of infrequent behaviors could be considered in our future research activity too.

On the other hand, the trace pre-processing approach proposed in [61] is more knowledge-intensive with respect to clustering, and requires the availability of additional information that we do not suppose to have in our traces.

In summary, the panorama on BP management in health care, to which we are applying our framework, is huge and very active. However, it is worth noting that our work is absolutely general. Indeed, we plan to test it in non-medical domains as well.

6. Concluding remarks and future work

In this work, we have described a case retrieval and clustering approach to process change and analysis. In particular, we have defined a proper case structure and a new distance measure, that are exploited to retrieve traces similar to the current one. Our system also allows to automatically cluster the trace database content by resorting to hierarchical clustering techniques.

We believe that such functionalities can help end users who need to adapt a process instance to some unforeseen situation, by retrieving changes applied in the past to other instances of the same process. Moreover, process engineers can take advantage of the retrieval and clustering results for identifying the most frequent changes to the same process schema. Such changes can be an index of non-conformance of process executions with respect to proper constraints, but can also be a suggestion for properly revising an incorrect or obsolete process schema definition.

The experimental results presented in this paper testify the advantages of adopting a similarity metric which explicitly takes into account temporal information, and show the potential usefulness of the tool in the stroke management domain, in which we are conducting our first evaluations. Additional tests will be performed in the future; applications to different domains will be considered as well.

We are also addressing the problem of retrieval on large databases. Indeed, retrieval can become computationally expensive in these situations, as it has been highlighted in the literature (see e.g. [37], which specifically refers to BP management applications). To this end, we have implemented a non-exhaustive search strategy, that exploits a Pivoting-Based Retrieval technique, allowing one to focus on particularly promising regions of the search space, and to neglect the others. The experimental results we have collected so far are very encouraging too.

Nonetheless, our framework still presents some limitations.

First, we have implemented clustering in order to support trace analysis, and (if needed) trace redesign. However, at the moment we do not provide an automatic interface to analysis tools, such as logic-based conformance checking facilities. Process engineers can visually analyze clustering results, and manually input trace data to process analysis tools. In the future, we would like to overcome this limitation. Indeed, we are working at the incorporation of our work as a set of plug-ins in the ProM framework [62], which is an open source environment for process mining. Within ProM, our plug-ins will be made available for cooperation with a set of verification plug-ins (Woflan analysis, verification of Linear Temporal Logic formulas on a log, check of conformance between a given process model and a trace), and performance analysis plug-ins (basic statistical analysis and performance analysis with a given process model), already embedded in the environment. Through such interactions, we believe that our work will globally support a principled reengineering activity, in line with the objectives described in the Introduction.

Moreover, within ProM, clustering could also support the process mining task itself, improving the quality of the process mining output (as in e.g. [43,42], see Section 5).

Another limitation is related to the assessment of the quality of the available traces. By now, we do not perform any evaluation of trace quality, but make all of them available for retrieval and clustering. On the other hand, as highlighted in [17], traces have to be fully trustworthy, if one wants to obtain meaningful analysis results. Indeed, in the stroke application we could count on real world traces of level ⁎⁎⁎ [17] at least, i.e., it was guaranteed that the actions in the traces always matched reality. However, this might not be the case in different applications: sometimes the logging might be incomplete, or not fully corresponding to reality. If we want to plan experiments in other domains, we will have to address this issue as well.

Moreover, it is worth observing that our case structure is a very simple one, as we deal with traces, where only sequence is represented as a control flow relation. It might be very interesting to extend or framework in order to deal with a more complex (e.g., graph-based) case structure, as it happens in e.g. [14,37,63]. Studying this extension will be one of our future goals as well.

Additionally, our tool could support the retrieval of similar traces in systems for recommendations on next process steps (see e.g. [64]). These tools are very interesting, because they offer an on-line support for process execution, and are based on traces—instead of process schemata, which can be difficult to acquire/mine.

Furthermore, our approach could be properly adapted in order to cluster change logs, as suggested in [65]. Change logs are quite different from execution traces, as they record only changes with respect to the default process schema (and not all the flow). On the other hand, contextual information is sometimes available. This adaptation would thus probably rise some issues, however, it seems to us an interesting research direction, which we will consider in our future work.

References

[1] Workflow Management Coalition 〈http://www.wfmc.org/wfmc-publications.html〉.

[2] W.V. der Aalst, A. ter Hofstede, M. Weske, Business process management: a survey, in: Proceedings of the International Conference on Business Process Management (BPM), Lecture Notes in Computer Science, vol. 2678, Springer, 2003, pp. 1–12.

[3] P. Heimann, G. Joeris, C. Krapp, B. Westfechtel, Dynamite: dynamic task nets for software process management, in: Proceedings of the International Conference of Software Engineering, Berlin, 1996, pp. 331–341.

[4] B. Weber, W. Wild, Towards the agile management of business processes, in: K.D. Althoff, A. Dengel, R. Bergmann, M. Nick, T. Roth-Berghofer (Eds.), Professional Knowledge Management WM 2005, Lecture Notes in Computer Science, vol. 3782, Springer, Berlin/Washington, DC, 2005, pp. 409–419.

[5] M. Reichert, P. Dadam, Adeptflex-supporting dynamic changes of workflows without losing control, Journal of Intelligent Information Systems 10 (1998) 93–129.

[6] S. Rinderle, M. Reichtert, P. Dadam, Correctness criteria for dynamic changes in workflow systems—a survey, Data and Knowledge Engineering 50 (2004) 9–34.

[7] B. Weber, M. Reichert, W. Wild, Case-based maintenance for CCBR-based process evolution, in: T. Roth-Berghofer, M. Goker, H.A. Guvenir (Eds.), Proceedings of the European Conference on Case Based Reasoning (ECCBR) 2006, Lecture Notes in Artificial Intelligence, vol. 4106, Springer, Berlin, 2006, pp. 106–120.

[8] F. Casati, S. Ceri, B. Pernici, G. Pozzi, Workflow evolutions, Data and Knowledge Engineering 24 (1998) 211–238.

[9] A. Aamodt, E. Plaza, Case-based reasoning: foundational issues, methodological variations and systems approaches, AI Communications 7 (1994) 39–59.

[10] J. Surma, K. Vanhoof, Integration rules and cases for the classification task, in: M. Veloso, A. Aamodt (Eds.), Proceedings of the 1st International Conference on Case-Based Reasoning, Lecture Notes in Computer Science, vol. 1010, Springer, Sesimbra, Portugal, 1995, pp. 325–334.

[11] T. Madhusudan, J. Zhao, B. Marshall, A case-based reasoning framework for workflow model management, Data and Knowledge Engineering 50 (2004) 87–115.

[12] Z. Luo, A. Sheth, K. Kochut, J. Miller, Exception handling in workflow systems, Applied Intelligence 13 (2000) 125–147.

[13] B. Weber, M. Reichert, S. Rinderle-Ma, W. Wild, Providing integrated life cycle support in process-aware information systems, International Journal of Cooperative Information Systems 18 (2009) 115–165.

[14] M. Minor, A. Tartakovski, D. Schmalen, R. Bergmann, Agile workflow technology and case-based change reuse for long-term processes, International Journal of Intelligent Information Technologies 4 (1) (2008) 80–98.

[15] S. Montani, Prototype-based management of business process exception cases, Applied Intelligence, http://dx.doi.org/10.1007/s10489-009-0165-z (published online in February 2009).

[16] W.V. der Aalst, Process Mining. Discovery, Conformance and Enhancement of Business Processes, Springer, 2011.

[17] IEEE Taskforce on Process Mining: Process Mining Manifesto 〈http://www.win.tue.nl/ieeetfpm〉.

[18] L.T. Ly, S. Rinderle, P. Dadam, Integration and verification of semantic constraints in adaptive process management systems, Data & Knowledge Engineering 64 (1) (2008) 3–23.

[19] A. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, Soviet Physics Doklady 10 (1966) 707–710.

[20] L. Yujian, L. Bo, A normalized levenshtein distance metric, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 1085–1091.

[21] J. Allen, Towards a general theory of action and time, Artificial Intelligence 23 (1984) 123–154.

[22] A. Lanz, B. Weber, M. Reichert, Workflow time patterns for process-aware information systems, in: Proceedings of the BMMDS/EMM-SAD, 2010, pp. 94–107.

[23] S. Kurtz, Approximate string searching under weighted edit distance, in: Proceedings of the 3rd South American Workshop on String Processing, Carlton University Press, Recife, 1996.

[24] M. Palmer, Z. Wu, Verb semantics for English–Chinese translation, Machine Translation 10 (1995) 59–92.

[25] R. Bergmann, A. Stahl, Similarity measures for object-oriented case representations, in: B. Smyth, P. Cunningham (Eds.), Proceedings of the European Workshop on Case-Based Reasoning (EWCBR) 1998, Lecture Notes in Artificial Intelligence, vol. 1488, Springer-Verlag, Berlin, 1998.

[26] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: Proceedings of the IJCAI, 1995, pp. 448–453.

[27] R. Page, M. Holmes, Molecular Evolution: A Phylogenetic Approach, Wiley, 1998.

[28] A. Marzal, E. Vidal, Computation of normalized edit distance and applications, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (1993) 926–932.

[29] C. Freska, Temporal reasoning based on semi-intervals, Artificial Intelligence 54 (1992) 199–227.

[30] R. Sokal, C. Michener, A statistical method for evaluating systematic relationships, University of Kansas Science Bulletin 38 (1958) 1409–1438.

[31] D. Inzitari, G. Carlucci, Italian stroke guidelines (spread): evidence and clinical practice, Neurological Sciences 27 (2006) s225–s227.

[32] A. Yip, T. Chan, T. Mathew, A Scale Dependent Model for Clustering by Optimization of Homogeneity and Separation, CAM Technical Report 03-37, Department of Mathematics, University of California, Los Angeles, 2003.

[33] R. Sharan, R. Shamir, CLICK: a clustering algorithm for gene expression analysis, in: Proceedings of the International Conference on Intelligent Systems for Molecular Biology, 2000, pp. 260–268.

[34] R. Duda, P. Hart, D. Stork, Pattern Classification, Wiley-Interscience, New York, 2001.

[35] P. Francis, D. Leon, M. Minch, A. Podgurski, Tree-based methods for classifying software failures, in: International Symposium on Software Reliability Engineering, IEEE Computer Society, 2004, pp. 451–462.

[36] S. Montani, G. Leonardi, A case-based approach to business process monitoring, in: M. Bramer (Ed.), Proceedings of the World Computer Congress—International Federation for Information Processing (IFIP), Springer-Verlag, Berlin, 2010, pp. 101–110.

[37] R. Bergmann, Y. Gil, Retrieval of semantic workflows with knowledge intensive similarity measures, in: A. Ram, N. Wiratunga (Eds.), Proceedings of the International Conference on Case-Based Reasoning (ICCBR) 2011, Lecture Notes in Artificial Intelligence, vol. 6880, Springer-Verlag, Berlin, 2011.

[38] R. Socorro, L. Mico, J. Oncina, A fast pivot-based indexing algorithm for metric spaces, Pattern Recognition Letters 32 (2011) 1511–1516.

[39] L. Portinale, P. Torasso, D. Magro, Selecting most adaptable diagnostic solutions through pivoting-based retrieval, in: D. Leake, E. Plaza (Eds.), Proceedings of the 2nd International Conference on Case-Based Reasoning, Lecture Notes in Computer Science, vol. 1266, Springer, Providence, RI, USA, 1997, pp. 393–402.

[40] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, University of California Press, 1967, pp. 281–297.

[41] A. Beygelzimer, S. Kakade, J. Langford, Cover trees for nearest neighbor, in: Proceedings of the International Conference on Machine learning, ACM, New York, NY, USA, 2006, pp. 97–104.

[42] G. Greco, A. Guzzo, L. Pontieri, D. Sacca, Discovering expressive process models by clustering log traces, IEEE Transactions on Knowledge and Data Engineering 18 (2006) 1010–1027.

[43] R.J.C. Bose, W.V. der Aalst, Context aware trace clustering: towards improving process mining results, in: Proceedings of the SIAM International Conference on Data Mining, Springer, 2009, pp. 401–412.

[44] W.V. der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm, A. Weijters, Workflow mining: a survey of issues and approaches, Data and Knowledge Engineering 47 (2003) 237–267.

[45] L. van Elst, F. Aschoff, A. Berbardi, H. Maus, S. Schwarz, Weakly-structured workflows for knowledge-intensive tasks: an experimental evaluation, in: Proceedings of the 12th IEEE International Workshops on Enabling Technologies (WETICE), Infrastructure for Collaborative Enterprises, IEEE Computer Society, Los Alamitos, 2003, pp. 340–345.

[46] H. Bunke, B. Messmer, Similarity measures for structured representations, in: Proceedings of the European Workshop on Case-based Reasoning (EWCBR), Lecture Notes in Computer Science, vol. 837, Kaiserslautern, 1993, pp. 106–118.

[47] S. Kapetanakis, M. Petridis, B. Knight, J. Ma, L. Bacon, A case based reasoning approach for the monitoring of business workflows, in: I. Bichindaritz, S. Montani (Eds.), Proceedings of the International Conference on Case Based Reasoning (ICCBR), Springer, Berlin, 2010, pp. 390–405.

[48] C. Combi, M. Gozzi, B. Oliboni, J. Juarez, R. Marin, Temporal similarity measures for querying clinical workflows, Artificial Intelligence in Medicine 46 (2009) 37–54.

[49] A. Rozinat, W.V. der Aalst, Conformance checking of processes based on monitoring real behavior, Information Systems 33 (2008) 64–95.

[50] J. Swinnen, B. Depaire, M.J. Jans, K. Vanhoof, A process deviation analysis—a case study, in: Proceedings of the Business Process Management Workshops, vol. 1, 2012, pp. 87–98.

[51] R.J.C. Bose, W.V. der Aalst, Trace alignment in process mining: opportunities for process diagnostics, in: Proceedings of the 8th International Conference on Business Process Management (BPM'10), Springer, 2010, pp. 227–242.

[52] R. Mans, M. Schonenberg, M. Song, W.V. der Aalst, P. Bakker, Application of process mining in healthcare—a case study in a dutch hospital, in: Biomedical Engineering Systems and Technologies, Communications in Computer and Information Science, vol. 25, Springer, 2009, pp. 425–438.

[53] A. Rebuge, D. Ferreira, Business process analysis in healthcare environments: a methodology based on process mining, Information Systems 37 (2012) 99–116.

[54] R.J.C. Bose, W.V. der Aalst, I. Zliobaite, M. Pechenizkiy, Handling concept drift in process mining, in: Proceedings of the CAiSE, 2011, pp. 391–405.

[55] T. Stocker, Time-based trace clustering for evolution-aware security audits, in: Proceedings of the Business Process Management Workshops, vol. 2, 2012, pp. 471–476.

[56] F. Caron, J. Vanthienen, J.D. Weerdt, B. Baesens, Advanced care-flow mining analysis, in: Proceedings of the Business Process Management Workshops, vol. 1, 2012, pp. 167–168.

[57] C. Reuter, P. Dadam, S. Rudolph, W. Deiters, S. Trillisch, Guarded process spaces (GPS): a navigation system towards creation and dynamic change of healthcare processes from the end-user's perspective, in: Proceedings of the Business Process Management Workshops, vol. 2, 2012, pp. 237–248.

[58] M. Grando, W.V. der Aalst, R. Mans, Reusing a declarative specification to check the conformance of different cigs, in: Proceedings of the Business Process Management Workshops, vol. 2, 2012, pp. 188–199.

[59] A. Bottrighi, F. Chesani, P. Mello, M. Montali, S. Montani, P. Terenziani, Conformance checking of executed clinical guidelines in presence of basic medical knowledge, in: Proceedings of the Business Process Management Workshops, vol. 2, 2012, pp. 200–211.

[60] R. Mans, H. Schonenberg, G. Leonardi, S. Panzarasa, A. Cavallini, S. Quaglini, W.V. der Aalst, Process mining techniques: an application to stroke care, in: S. Andersen, G.O. Klein, S. Schulz, J. Aarts (Eds.), Proceedings of the MIE. Studies in Health Technology and Informatics, vol. 136, IOS Press, 2008, pp. 573–578.

[61] R.J.C. Bose, W.V. der Aalst, Analysis of patient treatment procedures, in: Proceedings of the Business Process Management Workshops, vol. 1, 2012, pp. 165–166.

[62] B. van Dongen, A.A. De Medeiros, H. Verbeek, A. Weijters, W.V. der Aalst, The proM framework: a new era in process mining tool support, in: G. Ciardo, P. Darondeau (Eds.), Knowledge Management and its Integrative Elements, Springer, 2005, pp. 444–454.
[63] J. Kendall-Morwick, D. Leake, A toolkit for representation and retrieval of structured cases, in: Proceedings of the ICCBR-11 Workshop on Process-Oriented Case-Based Reasoning, 2011.
[64] C. Haisjackl, B. Weber, User assistance during process execution—an experimental evaluation of recommendation strategies, in: M. zur Muehlen, J. Su (Eds.), Business Process Management Workshops, LNBIP, vol. 66, Springer, Berlin, 2011, pp. 134–145.
[65] C.W. Guenther, S. Rinderle-Ma, M. Reichert, W.V. der Aalst, J. Recker, Using process mining to learn from process changes in evolutionary systems, International Journal of Business Process Integration and Management 3 (1) (2008) 61–78.