Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Viale Teresa Michel 11, 15121 Alessandria
http://www.di.unipmn.it

Trace retrieval and clustering for business process monitoring

G. Leonardi, S. Montani (giorgio.leonardi@mfn.unipmn.it, stefania.montani@mfn.unipmn.it)

TECHNICAL REPORT TR-INF-2012-03-01-UNIPMN
(March 2012)

The University of Piemonte Orientale Department of Computer Science Research Technical Reports are available via WWW at URL `http://www.di.unipmn.it/`. Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

2011-04 *Achieving completeness in bounded model checking of action theories in ASP*, L. Giordano, A. Martelli, D. Theseider Dupré, December 2011.

2011-03 *SAN models of a benchmark on dynamic reliability*, D. Codetta Raiteri, December 2011.

2011-02 *A new symbolic approach for network reliability analysis* , M. Beccuti, S. Donatelli, G. Franceschinis, R. Terruggia, June 2011.

2011-01 *Spaced Seeds Design Using Perfect Rulers*, L. Egidi, G. Manzini, June 2011.

2010-04 *ARPHA: an FDIR architecture for Autonomous Spacecrafts based on Dynamic Probabilistic Graphical Models*, D. Codetta Raiteri, L. Portinale, December 2010.

2010-03 *ICCBR 2010 Workshop Proceedings*, C. Marling, June 2010.

2010-02 *Verifying Business Process Compliance by Reasoning about Actions*, D. D'Aprile, L. Giordano, V. Gliozzi, A. Martelli, G. Pozzato, D. Theseider Dupré, May 2010.

2010-01 *A Case-based Approach to Business Process Monitoring*, G. Leonardi, S. Montani, March 2010.

2009-09 *Supporting Human Interaction and Human Resources Coordination in Distributed Clinical Guidelines*, A. Bottrighi, G. Molino, S. Montani, P. Terenziani, M. Torchio, December 2009.

2009-08 *Simulating the communication of commands and signals in a distribution grid*, D. Codetta Raiteri, R. Nai, December 2009.

2009-07 *A temporal relational data model for proposals and evaluations of updates*, L. Anselma, A. Bottrighi, S. Montani, P. Terenziani, September 2009.

2009-06 *Performance analysis of partially symmetric SWNs: efficiency characterization through some case studies*, S. Baarir, M. Beccuti, C. Dutheillet, G. Franceschinis, S. Haddad, July 2009.

2009-05 *SAN models of communication scenarios inside the Electrical Power System*, D. Codetta Raiteri, R. Nai, July 2009.

2009-04 *On-line Product Configuration using Fuzzy Retrieval and J2EE Technology*, M. Galandrino, L. Portinale, May 2009.

2009-03 *A GSPN Semantics for Continuous Time Bayesian Networks with Immediate Nodes*, D. Codetta Raiteri, L. Portinale, March 2009.

2009-02 *The TAAROA Project Specification*, C. Anglano, M. Canonico, M. Guazzone, M. Zola, February 2009.

Trace retrieval and clustering for business process monitoring

Stefania Montani, Giorgio Leonardi

DISIT, Institute of Computer Science, Università del Piemonte Orientale,
Viale Michel 11, I-15121, Alessandria, Italy
tel: +390131360158; fax: +390131360198; stefania.montani@unipmn.it

Abstract

The agile workflow technology deals with flexible workflow adaptation and overriding, in case of foreseen as well as unforeseen changes and problems in the operating business environment. One key issue that an agile workflow system should address is Business Process (BP) monitoring, which we define as a set of activities for organizing process instance logs and for highlighting non-compliances and adaptations with respect to the default process schema. Such activities can be the starting point for a set of a-posteriori log analyses.

In this paper, we describe an automated support to BP monitoring, 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 support both to end users, in the process instance execution phase, and to process engineers, in (formal) process quality evaluation and long term process schema redefinition.

Retrieval and clustering rely on a distance definition able to take into account temporal information in traces.

Keywords: Business Process Monitoring, Case-Based Retrieval, Hierarchical Clustering, Temporal Constraints

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 the workflow technology [1, 2].

BP optimization, in particular, may ask the enterprise to be able to flexibly change and adapt the 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] is the technical solution which has been invoked to deal with such adaptation and overriding needs. It can support both ad-hoc changes of individual process instances [5, 6], operated by end users, and modifications at the general process schema level, operated by process engineers - applicable even if the default schema is already in use by some running instances [5, 7].

In order to provide an effective and quick workflow change 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) [8] 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). 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 cases have often been resorted to in order to describe exceptions, in various domains (see e.g. [9]), and many examples of CBR-based process change reuse and workflow adaptation support have been proposed (see e.g. [10, 11, 6, 12, 13]).

A very critical issue to be addressed in agile workflow systems is the one of *BP monitoring* (see e.g. [14]), which we define as a set of activities aiming at organizing process instance logs, and at highlighting possible non-compliances with respect to the default process schema, thus serving as the starting point for a-posteriori log analyses. Providing BP monitoring functionality is non trivial. As a matter of fact, deviations from a default process schema generate instance logs, typically stored as traces of actions, that are different from how they were supposed to be. Often no contextual information, which could justify the reasons for deviation, is recorded in the traces themselves. A facility able to intelligently exploit traces of process executions, by retrieving similar ones, and by automatically organizing them, would then be an added value for an agile workflow tool.

In this paper, we propose an approach to BP monitoring which adopts the **retrieval** step of the CBR methodology, in order to look for cases (i.e. traces) similar to the current one. Our framework also supports an automatic organization of the case base content (i.e. of all available traces) through the application of hierarchical **clustering** techniques.

Retrieval can provide support to *end users* in the process instance execution phase, especially when dealing with atypical situations. Indeed, suggestions on how to modify 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.

Moreover, clustering can help *process engineers* in quality evaluation (e.g. as an input to a formal verification of the conformance of traces to proper semantic constraints [15]). 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 redefine process schemas, in order to incorporate the most frequent and significant changes once and for all.

Retrieval and clustering rely on a distance definition able to take into account temporal information, i.e. quantitative as well as qualitative temporal constraints, in traces. Technical details of the approach are presented in section 2.

Our approach is currently being experimented in the field of stroke management; experimental results are described in section 3.

Section 4 addresses some comparisons with related works; finally, section 5 is devoted to conclusions and future research directions.

2. A framework for supporting BP monitoring

Methodological and technical details on the realization of our framework are described in the following. In particular, both retrieval and clustering strongly rely on the notion of case, and on case distance definition, which 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 an instance of a given process schema. In particular, every trace is a sequence of actions, each one stored

with its execution starting and ending times. The actions timestamps also allow to derive information about action durations, as well as qualitative (e.g. Allen's *before* [16]) and quantitative temporal constraints (e.g. delay length) between pairs of consecutive actions.

Case *distance* is then calculated on the basis of:

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

Operatively, we first take into account action types, by calculating a modified edit distance which we have called **trace edit distance**. The minimization of trace edit distance provides the optimal alignment of two traces.

Given the alignment, we can take into account temporal information. In particular, we compare the durations of aligned actions by means of a metric known as **interval distance**.

Moreover, we take into account the temporal constraints between two pairs of corresponding 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 (e.g. by calculating interval distance between the two overlap lengths).

These three contributions (i.e. minimal trace edit distance, interval distance between durations, neighbors-graph distance or interval distance between pairs of actions) are finally combined in an additive way (a weighted sum may be used as well).

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 consider 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.

In our approach, the cost of a *substitution* is not always set to 1, as in the classical edit distance. In fact, as in the weighted edit distance (see e.g. [17]), 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 more two actions are close in the taxonomy, the less penalty has to be introduced for substitution ([18]; see also [19, 20, 21]). In detail, in our work substitution penalty is set to the *taxonomic distance dt* between the two actions ([18], see Definition 2 below), i.e. to the normalized number of arcs on the path between the two actions in the taxonomy.

Insertions do not always cost 1 as well. In fact, the insertion of one (or of a few) action(s) may sometimes introduce a (minor) change in a specific trace with respect to a reference trace, without changing the overall semantics/goals of the sequence of actions being considered. Our definition allows to capture this situation, by distinguishing between *indirections* (i.e. insertions of one or more actions within the trace, otherwise very similar to the reference one - see figure 1, third case), and insertions in the head/tail portion of the trace, which becomes a superstring of the reference one. Recognizing indirections can be very relevant for many BP monitoring applications, e.g. for medical ones[1]. Our definition introduces a knowledge-based parametrized weight $\in [0, 1]$ for indirections, which depends on the action type. The final penalty of an indirection is therefore equal to 1 multiplied by the weight. Naturally, the more actions are introduced, the more indirection is obtained in the path, and the more penalties are added. *Deletions* simply work dually with respect to insertions.

The trace edit distance $trace_{ed}(P, Q)$ between two traces P and Q is finally defined as the total cost of the set of edit operations which transform one trace into the other.

Formally, we provide the following definitions:

Definition 1: Trace edit distance. Let P and Q be two traces of actions, and let α and β be two actions. The *trace edit distance* between P and Q is

[1]Consider, for instance, two hospitals applying the same care protocol to their patients. Suppose that, in the first hospital, an additional action, such as blood pressure measurement, is always executed at 5 PM on all patients, including the ones following the protocol at hand. Such action will be identified as an insertion with respect to the traces collected at the second hospital: however, certainly it does not represent a major change. Our system manages it as an indirection.

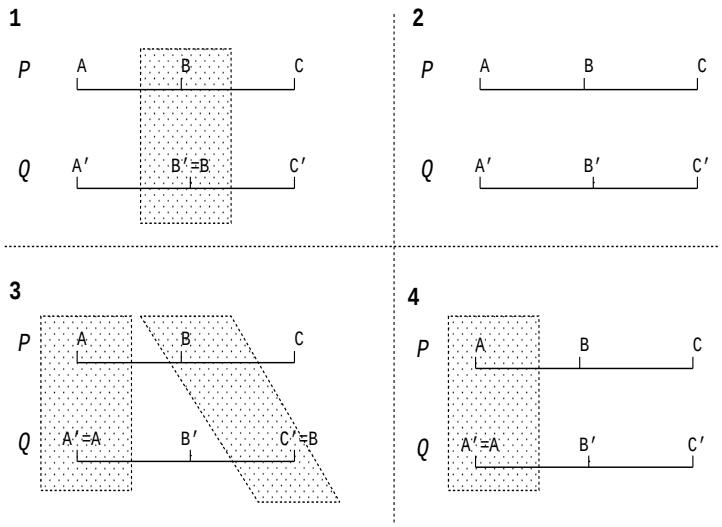

Figure 1: Comparison of the i-th action B in trace P with the j-th action B' in trace Q. If the two actions are comparable (see terminology below - first case in the figure), the distance is not increased (or minimally increased, if the actions are not identical), and the next actions are considered. Otherwise, in order to discover insertions (and specifically indirections), we also compare the $(i-1)$-th action A in P with the $(j-1)$-th action A' in Q. If their are not comparable (second case in the figure), no indirection is being generated, and $dt(B, B')$ is added to the distance calculation. On the other hand, if A and A' are comparable, we need to compare B with the $(j+1)$-th action in Q: if they are comparable as well (third case in the figure), B' has been inserted creating an indirection on trace Q, and the cost $1 * w_{B'}$ must be added to the trace edit distance calculation; otherwise (fourth case in the figure), a cost equal to $dt(B, B')$ will be added.

defined as:

$$trace_{ed}(P, Q) = \sum_{i=1}^{k} c(e_i)$$

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

– $c(e_i) = dt(\alpha, \beta)$, if e_i is the substitution of α (appearing in P) with β (appearing in Q), with $dt(\alpha, \beta)$ defined as in Definition 2 below;

– $c(e_i) = 1 * w_\alpha$, if e_i is the insertion (the deletion) of action α in P (from Q), with w_α defined as in Definition 3 below.

Definition 2: 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* 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[2].

Terminology. Two actions α and β are **comparable** if $dt(\alpha, \beta) \leq \tau$, where τ is a threshold to be set on the basis of domain knowledge.

Definition 3: Action weight. Let P and Q be two traces of actions and let α be an action (appearing in Q). The *action weight* w_α of α is defined as:

– $w_\alpha \in [0, 1]$ - to be set on the basis of domain knowledge - if α generates an indirection (see figure 1, third case) in Q with respect to P;

– $w_\alpha = 1$ otherwise (e.g. if P is a substring of Q, and α is an action in the head or tail portion of Q, not matched to any action in P).

An example of trace edit distance calculation is illustrated in figure 1 (for the sake of clarity, an indirection due to a single action is considered in the figure).

The minimal value of trace edit distance allows to find the optimal alignment between two traces being compared.

[2]Note that, if $dt(\alpha, \beta) > th$, being $th \in [0, 1]$ a proper, domain-dependent threshold, $dt(\alpha, \beta)$ can be forced to 1.

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 [16] *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 (e.g. the durations of two aligned actions, the extensions of the delays between two pairs of aligned actions), we resort to the interval distance (see [14]), defined as follows:

Definition 4: Interval distance. Let i and j be two intervals of length len_i and len_j respectively. The *interval distance* between i and j is defined as:

$$interval_d(i,j) = \frac{|len_i - len_j|}{|len_i| + |len_j|}$$

Note that, in the case of two instantaneous actions, the interval distance between their durations is set to 0 [14].

In some situations (corresponding to insertions/deletions), it may happen that a specific action in a trace has no matching action in the other trace. Consider e.g. figure 1, third case: action A in trace P is aligned to action A' in trace Q; action B in P is aligned to action C' in Q, and B' has no matching action in trace P, since it is an insertion (indirection). Such a situation is managed by calculating interval distance just referring to matching actions. In the example, we thus calculate interval distance between the A - B delay, and the sum of the A' - B' delay, the B' - C' delay, and the duration of B' (which globally corresponds to the A' - C' delay in trace Q).

2.1.3. Neighbors-graph distance

When comparing two pairs of corresponding actions (e.g. A and B in trace P; the aligned actions A' and B' in trace Q), it may happen that they don't share the same qualitative constraint (e.g. A and B overlap in trace P; A' is before B' in trace Q). In this case, we cannot resort to interval distance to compare the inter-action constraints, because they have a different semantic meaning. On the other end, we can quantify the difference between the two

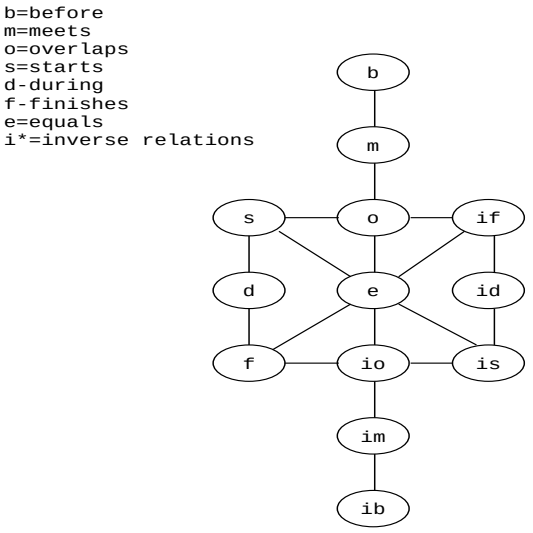

Figure 2: The *A-neighbors graph* proposed by Freska [22].

qualitative constraints by resorting to the *A-neighbors graph* proposed by Freska [22] (see figure 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 [16], and let G be the A-neighbors graph in figure 2. The *neighbors-graph distance* between i and j is defined as:

$$ngraph_d(i,j) = \frac{path(i,j,G)}{max(path(.,.,G))}$$

where $path(i,j,G)$ measures the shortest path on G between i and j, and $max(path(.,.,G))$ normalizes the distance considering the longest path on G.

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) [23]. UPGMA is typically applied in bioinformatics, where sequences of symbols (similar to our traces of actions) 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 the BP monitoring domain.

3. Experimental results

Since some experiments on our retrieval facility were already presented in [24], in this paper we will concentrate on clustering.

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 form of process logs (i.e. traces of execution), 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 BP monitoring techniques [25], like the ones introduced in this paper. These considerations motivated the choice of a medical application.

In particular, in our experiments we aimed at verifying whether the distance function described in this paper was able to overcome the performances of classical edit distance, and of previous versions of our distance definition we introduced in [24] and [26] respectively. Namely, in [24] we extended classical edit distance to trace edit distance. In [26] we also considered action durations and delays between actions, but we were unable to treat qualitative constraints other than *before* and *meets*, and we were unable to make comparisons between different qualitative constraints.

The hypothesis we wished to test was the following: including domain knowledge (through trace edit distance) improves clustering performances; including temporal information provides a further improvement (moreover, obviously, the use of neighbors-graph distance allows to deal with all kinds of traces, while only strictly sequential traces could be treated by the distance definition in [26]).

Clustering performances were evaluated by studying the capability of clearly isolating anomalous situations, and of refining cluster composition (to this end, we resorted to *homogeneity* [27], see section 3.2).

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 [28] 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 (CAT) 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. At the same time, administrative patient admission procedures must be fulfilled. 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.

3.2. Comparing different distance measures

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

- (1) edit distance;
- (2) trace edit distance;
- (3) a distance measure able to deal with atemporal information, through trace edit distance, as well as with action durations and delays between actions, through interval distance (see [26]). Such a distance is unable to treat qualitative constraints other than *before* and *meets*, and is unable to compare different qualitative constraints;
- (4) the distance measure introduced in section 2.1.

We made our experiments working on two databases:

- DB1, in which some traces were properly 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.

Both databases contained 100 traces.

Figure 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 figure 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 figure 3, lower part) 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. [27, 29, 30, 31]). A classical definition of cluster homogeneity is the following [27]:

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

where $|C|$ is the number of elements in cluster C, and $1 - dist(x,y)$ 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. [31]).

The average of the homogeneity H of the individual clusters can then 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 [27]. 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 by 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.

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

Specifically, clusters obtained by using distance (1) had an average homogeneity of 0.47 at level 3 of the hierarchy, while by 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).

Figure 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 distance (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, figure 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.

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 drug provision instead of antiaggregant drug provision). Distance (1) penalizes this difference, and early isolates the trace; on the other hand, distance (2) doesn't, 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

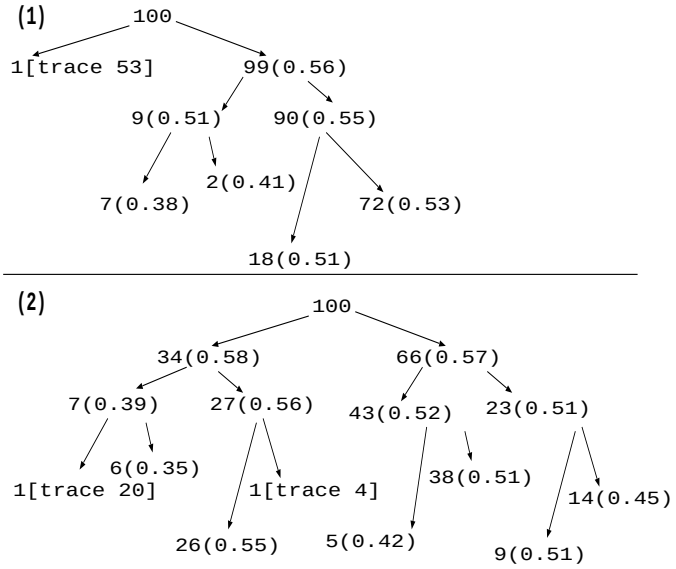

Figure 3: Part of the cluster hierarchies obtained by applying distance (1) (top) and distance (2) (bottom) on DB1. Every node represents a cluster and reports the number of traces in the cluster itself, and their average normalized edit distance (in brackets).

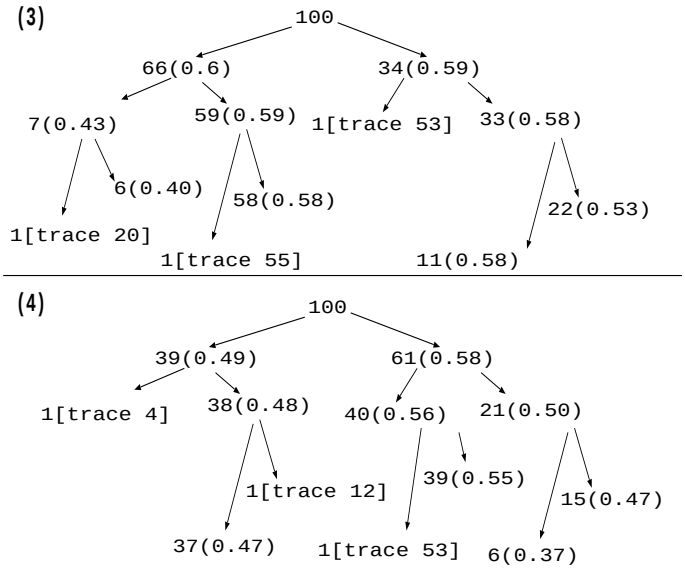

Figure 4: Part of the cluster hierarchies obtained by applying distance (3) (top) and distance (4) (bottom) on DB1.

correct, 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). 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. 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) doesn't. 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 graph; this contribution, therefore, does not penalize trace no. 20 as much as interval distance did.

In conclusion, such 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. Moreover, anomalous traces are correctly isolated.

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.

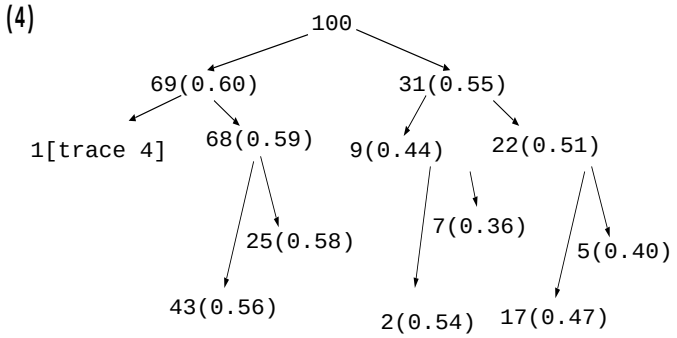

Figure 5: Part of the cluster hierarchy obtained by applying distance (4) on DB2.

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%

4. Related works

Examples of CBR tools in BP management, and specifically in process change support, are described in the literature (e.g. [10, 11, 6, 12, 13]); a few works exploiting clustering techniques are reported as well (e.g. [32, 33, 34]) - even though they mainly deal with process mining [35].

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 focus on this issue, and on the papers providing interesting solutions in relation to it.

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 [36], or conversational knowledge [4, 6]. 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 [37] has been proposed and adapted for similarity assessment in workflow change reuse [12].

Our approach somehow moves from the same graph edit distance definition. However, with respect to the work in [12], 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 [12].

On the other hand, when focusing on linear traces our approach is more general and flexible. As a matter of fact, we resort to taxonomic knowledge for comparing pairs of actions, so that two different actions do not always have a zero similarity. Additionally, we are able to recognize an indirect path from two actions, and to properly weight the degree of indirection in a parametrized way. 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 [12].

On the other hand, a treatment of temporal information in trace distance calculation has been proposed in [14]. 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 definition which is very close to ours. Differently from what we do, however, the work in [14] 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 the clustering algorithm. The distance function in [14] 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 [38] 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 [22], as we do. However, in [38] the alignment problem is strongly simplified, as they only match actions with the same name. Our approach thus extends their work.

It is also worth citing the approach in [34], which adapts edit distance to trace clustering (as we do), allowing to automatically derive the cost of edit operations. In such a work, however, temporal information is not considered. Moreover, clustering is mainly resorted to for improving process mining (by clustering instances in advance to process mining, the authors succeed in obtaining less "spaghetti like" process mining results). Process monitoring is not addressed.

5. Conclusions and future work

In this work, we have described a case retrieval approach to BP monitoring. In particular, we have defined a proper case structure and a new distance measure, that are exploited to retrieve traces of execution 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.

By now we made tests on small database sizes. However, calculation of similarity (for retrieval and clustering) can become computationally expensive when moving to very large databases. This problem has already been highlighted in process instance databases [39]. To this end, in the future we plan to investigate techniques to avoid exhaustive search of similar traces, focusing on particularly promising regions of the search space and neglecting the others. Specifically, we plan to resort to pivoting-based techniques, such as the AESA algorithm (see e.g. [40]).

Remarkably, our work is also being incorporated as a set of plug-ins in the ProM tool [41], which is an open source framework for process mining. As it is well known, process mining [35] describes a family of a-posteriori analysis techniques exploiting the information recorded in traces of actions. This process information can be used to discover the underlying process schema, when no a priori model is available. Once the process schema is obtained, our facilities can be used to support an analysis of deviations from the process schema itself[3]. Clustering could also support the process mining task (see e.g. [34]).

Additionally, our tool could also support the retrieval of similar traces in systems for recommendations on next process steps (see e.g. [42]). These tools are very interesting, because they offer an on-line support for process execution, and are based on process logs - instead of process schemas, which can be difficult to acquire/mine.

Furthermore, our approach could be properly adapted in order to cluster change logs, as suggested in [43]. 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, $http://www.wfmc.org/wfmc-publications.html$, last accessed on october 6th 2009.

[2] W. V. der Aalst, A. ter Hofstede, M. Weske, Business process management: a survey, in: Proc. Int. Conference on Business Process Management (BPM), LNCS 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 International Conference of Software Engineering, Berlin, 1996, pp. 331–341.

[3]Interestingly, even when the default process schema is available, mining a(n alternative) process model from the available traces can help to find deviations from the normal flow, thus enlarging the applicability of ProM (and of our facility) to more real world situations.

[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, LNCS 3782, Springer, Berlin, Washington DC, 2005, pp. 409–419.

[5] M. Reichtert, S. Rinderle, P. Dadam, Adept workflow management system: Flexible support for enterprise-wide business processes, in: W. M. P. V. der Aalst (Ed.), Proc. of International Conference on Business Process Management (BPM) LNAI 2678, Springer, Berlin, 2003, pp. 370–379.

[6] B. Weber, M. Reichert, W. Wild, Case-based maintenance for CCBR-based process evolution, in: T. Roth-Berghofer, M. Goker, H. A. Guvenir (Eds.), Proc. European Conference on Case Based Reasoning (ECCBR) 2006, LNAI 4106, Springer, Berlin, 2006, pp. 106–120.

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

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

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

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

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

[12] 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.

[13] S. Montani, Prototype-based management of business process exception cases, Applied Intelligence (published online in February 2009- DOI 10.1007/s10489-009-0165-z).

[14] 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.), Proc. International Conference on Case Based Reasoning (ICCBR), Springer, Berlin, 2010, pp. 390–405.

[15] 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.

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

[17] S. Kurtz, Approximate string seraching under weighted edit distance, in: Proc. 3rd South American Workshop on String Processing, Carlton University Press, Recife, 1996.

[18] M. Palmer, Z. Wu, Verb Semantics for English-Chinese Translation, Machine Translation 10 (1995) 59–92.

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

[20] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: Proc. IJCAI, 1995, pp. 448–453.

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

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

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

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

[25] 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.), Proc. MIE, Studies in Health Technology and Informatics 136, IOS Press, 2008, pp. 573–578.

[26] S. Montani, G. Leonardi, M. LoVetere, Case retrieval and clustering for business process monitoring, in: Proc. Process-Oriented Case-Based Reasoning Workshop, International Conference on Case Based Reasoning (ICCBR) 2011, Greenwich, 2011.

[27] 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.

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

[29] R. Sharan, R. Shamir, CLICK: A clustering algorithm for gene expression analysis, in: Proc. International Conference on Intelligent Systems for Molecular Biology, 2000, p. 260268.

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

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

[32] J. Jung, J. Bae, L. Liu, Hierarchical business process clustering, IEEE International Conf. on Services Computing 2 (2008) 613–616.

[33] M. Song, C. Gunther, W. V. der Aalst, Trace clustering in process mining, in: Business Process Management Workshops, 2008, pp. 109–120.

[34] R. J. C. Bose, W. V. der Aalst, Context aware trace clustering : towards improving process mining results, in: Proc. SIAM Int. Conference on Data Mining, Springer, 2000, pp. 401–412.

[35] 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.

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

[37] H. Bunke, B. Messmer, Similarity measures for structured representations, in: Proc. of the European Workshop on Case-based Reasoning (EWCBR), LNCS 837 , Kaiserslautern, 1993, pp. 106–118.

[38] 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.

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

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

[41] 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 Mangement and its Integrative Elements, Springer, 2005, pp. 444–454.

[42] 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 66, Springer, Berlin, 2011, pp. 134–145.

[43] 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.