Retrieval and Clustering for Business Process Monitoring: Results and Improvements

Stefania Montani and Giorgio Leonardi

DISIT, Sezione di Informatica, Università del Piemonte Orientale, Alessandria, Italy

Abstract. Business process monitoring is 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 typically serve as the starting point for a-posteriori log analyses.

In recent years, we have implemented a tool for supporting business process monitoring, which 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 clustering techniques. Retrieval and clustering rely on a distance definition able to take into account temporal information in traces.

In this paper, we report on such a tool, and present the newest experimental results.

Moreover, we introduce our recent research directions, that aim at improving the tool performances, usability and visibility with respect to the scientific community.

Specifically, we propose a methodology for avoiding exhaustive search in the trace database, by identifying promising regions of the search space, in order to reduce computation time.

Moreover, we describe how our work is being incorporated as a plug-in in ProM, an open source framework for process mining and process analysis.

1 Introduction

Business Process (BP) monitoring is 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 typically serve as the starting point for a-posteriori log analyses.

In recent years, we have implemented a BP monitoring framework [16, 18, 17], which we have so far applied to the field of stroke management[1]. Our tool supports *end users* (e.g. user physicians) in process instance modification, by retrieving traces of execution similar to the current one: suggestions on how to adapt 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. Additionally, our framework can automatically cluster the execution traces stored in the database on the basis

[1] However, our work is domain independent.

B. Díaz Agudo and I. Watson (Eds.): ICCBR 2012, LNCS 7466, pp. 269–283, 2012.

of their similarity, and allows *process engineers* (e.g. expert physicians, administrators) to inspect the obtained clusters, in order to visualize the most frequent changes. Indeed, since changes can be an indicator of non-compliance, clustering can be seen as the first step in a process quality evaluation activity, which can be realized by means of formal (e.g. logic-based) verification approaches. Moreover, 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 schemata, in order to incorporate the most frequent and significant changes once and for all, at least at the local level (e.g. referring to the hospital they work for).

In our framework, trace retrieval relies on the *retrieval* step of Case-Based Reasoning (CBR) [1]. As for clustering, we resort to a hierarchical clustering technique, known as Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [24]. Both retrieval and clustering exploit a distance definition able to take into account temporal information in traces. Technical details of the approach are summarized in section 2.

Our tool is being tested in the stroke management domain. In section 3 we report on some new experiments, specifically referring to the clustering functionality.

The results we have obtained up to now are very encouraging, but were drawn on quite a small trace database.

On the other hand, we are aware that search and calculation of similarity can become computationally expensive when working on very large databases. This problem has already been highlighted in process instance databases retrieval [4].

To this end, we are currently designing and implementing a methodology able to enhance the performances of our tool, avoiding exhaustive search of similar traces. Specifically, we are resorting to a *pivoting-based technique* (see e.g. [23, 20]), which allows one to focus on particularly promising regions of the search space, and to neglect the others. Details of this extension are described in section 4.

Moreover, in order to enhance usability, and to integrate our framework with powerful BP management tools already available in the literature, we are incorporating it as a set of plug-ins in the ProM framework [25]. ProM is an open source framework for process mining and process analysis. In ProM, once the process schema is learned, the incorporation of our facilities will help in the analysis of deviations from the process schema itself, which can be the input for a (formal) compliance verification, as observed above. Moreover, ProM offers process representation and visualization standards that can be helpful to improve usability and user-friendliness of our work. Finally, the implementation within ProM will make our work more visible, and available to the BP management scientific community. Technical details of the integration in ProM are presented in section 5.

Finally section 6 addresses some discussions and our concluding remarks.

2 A Framework for Supporting BP Monitoring

In our framework, trace retrieval relies on the *retrieval* step of Case-Based Reasoning (CBR) [1]. Indeed, CBR has been often resorted to in workflow systems with flexible adaptation capabilities, such as *agile workflow* systems [27]. CBR is in fact particularly well suited for managing exceptional situations, even when they cannot be foreseen or preplanned. Examples of CBR-based workflow tools can be found in e.g. [13–15]. In particular, in our work case 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.

As for clustering, we resort to a hierarchical clustering technique, known as Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [24]. UPGMA is typically applied in bioinformatics, where sequences of symbols (similar to our traces of actions) have to be compared. 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 some domains (e.g. medical ones).

Case retrieval and clustering strongly rely on the notion of case, and on case distance definition. Technically, 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. Actions timestamps also allow to derive information about action durations and (partial) overlaps/delays between 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**. In trace edit distance, the cost of a *substitution* is not always set to 1, as in the classical edit distance. 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, and substitution penalty is set to the normalized number of arcs on the path between the two actions in the taxonomy [16]. *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), and insertions in the head/tail portion of the trace, which becomes a superstring of the reference one. Recognizing such indirections can be very relevant for many BP monitoring applications (especially medical ones). Our definition introduces a knowledge-based parametrized weight $\in [0, 1]$, which depends on the action type. The final penalty of an indirection is set to 1 multiplied by the action weight. *Deletions* simply work dually with respect to

insertions. Trace edit distance between two traces is finally defined as the total cost of a sequence of edit operations which transform one trace into the other. Formal definitions can be found in [16]. The minimization of trace edit distance provides the optimal alignment of the 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** [18]. Interval distance calculates the normalized difference between the length of two intervals (representing action durations in this case).

Moreover, we are able to take into account the contribution 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 relations (e.g. A and B overlap in trace P; A' meets B' in trace Q, see [3]), by resorting to a metric known as **neighbors-graph distance** [17]. If the neighbors-graph distance is 0, because the two pairs of actions share the same qualitative relation (e.g. A and B overlap in trace P; A' and B' also overlap in trace Q), we compare the 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 more formal description of our framework can be found in [16, 18, 17].

3 Experimental Results

We presented some retrieval and clustering experimental results in our previous works [16, 18, 17]. All experiments were conducted in the field of stroke management.

Since then, we collected more real world stroke management traces. In this paper, we report on additional clustering experiments, conducted on these new data.

In particular, in our experiments we aimed at verifying whether the distance function summarized in section 2 was able to overcome the performances of a previous version we introduced in [16], which did not take into account temporal information. The hypothesis we wished to test was the following: including temporal information allows to better characterize clusters, leading to a higher cluster *homogeneity*.

Moreover, we report on an additional experimental study, in which we tried to enhance the clustering results by introducing a pre-processing step, meant to separate traces belonging to patients with different feature values.

The database on which we made our experiments was composed of 377 traces collected at one of the largest stroke management units in the Lombardia region, Italy.

3.1 Comparing Two Distance Measures

As a first experiment, we compared the clustering results obtained by adopting two different distance measures: the one summarized in this paper, and a former version [16], which did not take into account temporal information.

Figure 1 shows part of the cluster hierarchies we obtained. We can observe that the content of the resulting clusters at the various levels of the hierarchies is very different in the two situations.

In particular, the hierarchy built using the distance measure which does not consider temporal information (see figure 1, upper part) is very unbalanced: every node is split into two children, one of which corresponds to a very big cluster (containing most of the traces of its parent node), while the other contains just a few traces. However, an inspection of cluster contents, made with the help of the domain experts working with us, revealed that small clusters were not originated from the ability to quickly isolate anomalous situations, and that their content is typically not *homogeneous* (see below).

On the other hand, the hierarchy built resorting to the distance measure described in this paper (see figure 1, lower part) appears to be much more balanced, and every node is normally split into two clusters of more comparable dimensions.

A different view of the clustering output is shown in figure 2, where the upper part shows the hierarchy of clusters, while the lower part details the content of one cluster (highlighted in the upper part of the figure itself). Identical actions in different traces are in the same color. In this way it is quite straightforward to compare traces by visual inspection. Temporal constraints are rendered as well. For instance, trace 2 shows an *overlaps* example, while trace 8 shows a *during* example. Durations and delays are straightforwardly interpretable as well.

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. [28, 22, 9]). A classical definition of cluster homogeneity is the following [28]:

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

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.

A (weighted) 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 two different distance measures).

An appropriate definition of $dist(x,y)$ is problem dependent [28]. 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 computed the average of cluster homogeneity values level by level in the two hierarchies.

Clusters obtained by using the new metric had an average homogeneity of 0.41 at level 2 of the hierarchy, while with the metric in [16] they had an average homogeneity of 0.25. At level 3, the new metric led to an average homogeneity of 0.45, while the metric in [16] allowed to reach only a value of 0.28. Similar results were obtained if working at other intermediate levels of the hierarchy, with some clusters built using the old metric even reaching an average edit distance value of 0.88.

Such experiments show that the use of temporal information in the distance definition allows to obtain more homogeneous and compact cluster (i.e. able to aggregate closer examples) in the intermediate levels of the hierarchy, which is a desirable results and a meaningful outcome, especially in a domain in which the role of time is obviously central.

These results also confirm the outcomes we already obtained in our first experimental studies [18, 17], which were conducted on a smaller database.

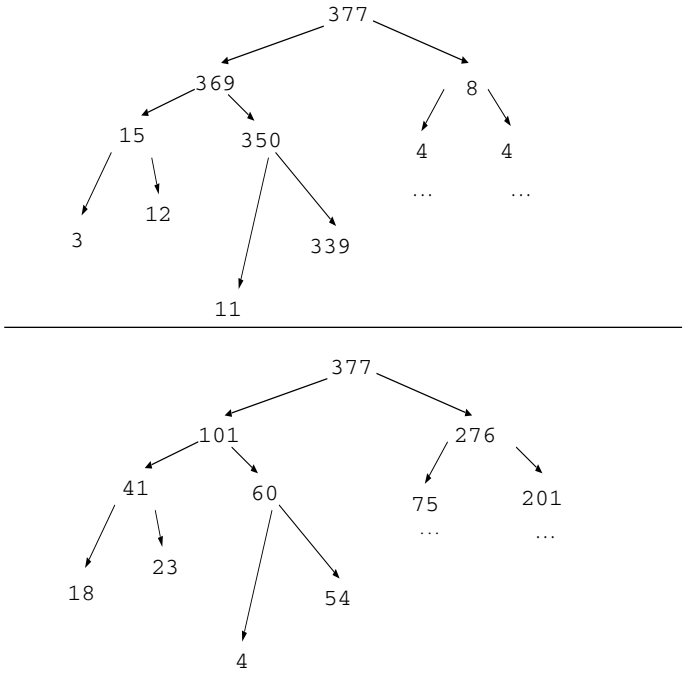

Fig. 1. Part of the cluster hierarchies obtained by applying the distance definition in [16] (top) and the one presented in this paper (bottom). Every node represents a cluster and reports the number of traces in the cluster itself.

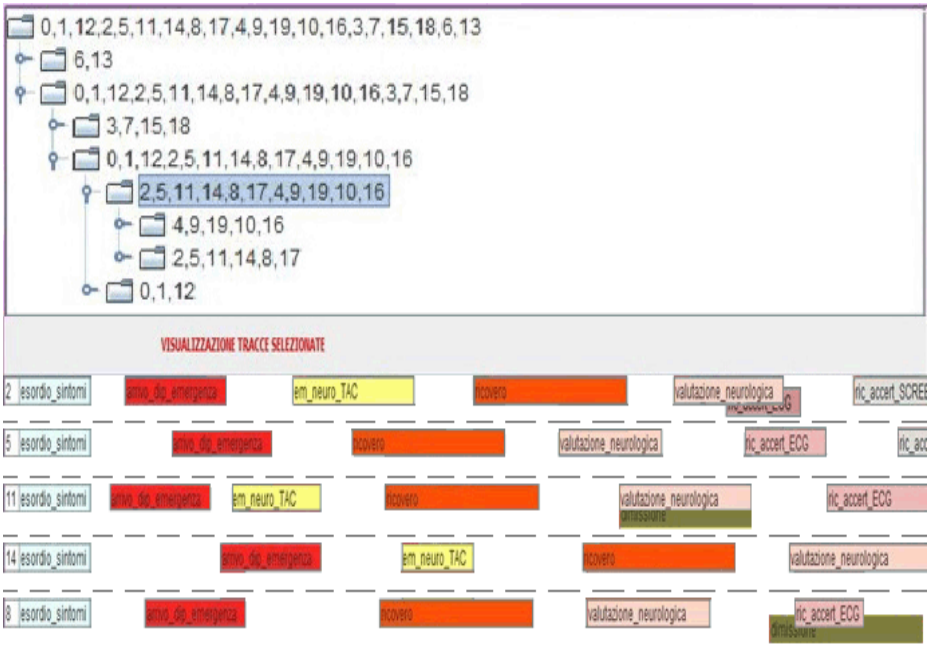

Fig. 2. A snapshot of clustering output as rendered by the system graphical interface

3.2 Verifying Care Delivery Similarity across Age Groups

Our second experiment was more oriented at the specific application needs, aiming at verifying the medical correctness of the stroke management procedures, in relation to patient age groups.

In particular, a recent cohort study [21] verified that, in the context of a province-wide coordinated stroke care system in Canada, stroke care delivery was similar across all age groups (even though, quite naturally, increasing age was associated with stroke severity and fatality). We aimed at verifying the absence of significant differences in stroke care delivery in different age groups in our database as well.

To this end, we implemented a pre-processing step, meant at separating traces belonging to patients with different anagraphical characteristics in different groups. In particular, we divided our database into 3 groups, filtering traces on the basis of the age of the patient they were applied to (i.e. ≤ 65, > 65 and ≤ 80, > 80). Traces in group ≤ 65 covered 25% of the total; traces in group > 80 covered 34%.

Observe that age is not explicitly recorded in traces, which just store the executed actions, and their temporal constraints. Indeed age is not used in our distance calculation - only in pre-processing.

We then performed intra-group clustering exploiting the distance measure summarized in this paper, and calculated cluster homogeneity.

Homogeneity values were comparable across the three classes, and comparable to the ones obtained on the whole database. At level 3 of the hierarchy, for instance, the average homogeneity was 0.43 in group ≤ 65, and 0.44 in group > 80 - not very different from the 0.45 value obtained on the entire database.

Such results highlight that intra-group treatments are not more homogeneous than inter-group ones; we believe that this can be seen as an indicator of the fact that care delivery is similar across all ages. In order to prove this statement, we plan to conduct a detailed evaluation of cluster contents with domain experts in the next months.

Along the same direction, in the future we will also pre-process our database by filtering traces on the basis of the National Institutes of Health Stroke Scale (NIHSS) values. The initial NIHSS score provides important prognostic information: approximately 60% to 70% of patients with an acute stroke and a baseline NIHSS score < 10 will have a favorable outcome after 1 year, as compared with only 4% to 16% of those with a score > 20 [2]. We plan to investigate if cluster homogeneity increases when applying a pre-processing step able to separate traces in groups corresponding to different NIHSS values: this could indicate that different procedures are applied to different groups of patients (which seems to be reasonable).

We will also evaluate whether to include these additional patient's features, which are not explicitly reported in traces, in the distance definition, in order to automatically include their treatment in the clustering process.

4 Improving Performances through a Pivoting-Based Technique

Retrieval time was very reasonable in the experiments we have conducted so far (see [16]). However, we were working on a small database. As the number of items in the database becomes higher and higher, the tool performances could progressively and significantly worsen. Nevertheless, computation time can be reduced, if a non-exhaustive search and distance calculation strategy is implemented.

The solution we are proposing is pivoting-based retrieval (PBR; see also [20, 23]).

The main idea in PBR consists in:

- computing the distance between a representative case and all the other cases (off-line);
- computing the distance between the representative case and the input case;
- estimating the distance between the input case and all the remaining cases by 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 figure 3). The following iterative procedure is then applied:

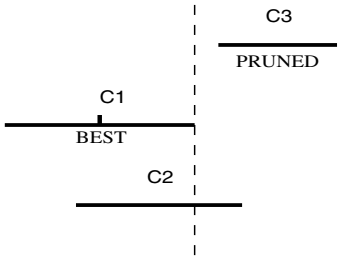

Fig. 3. Bound pruning in PBR

1. Initialization: $BESTp = \infty$ e $SOL = \{\ \}$
2. 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 $BEST = DIST$;
3. If $BESTp > BEST$ set $SOL = PIVOT$ and $BESTp = BEST$
4. Else if $BESTp = BEST$ set $SOL = \{PIVOT, SOL\}$
5. Prune the intervals whose lower bound is bigger than $BEST$, and remove the Pivot from the set of cases (see figure 3)
6. Back to step 2.

The choice of the most suitable representative case in the initialization phase is crucial to speed up the algorithm.

We have made some first tests by defining the representative case 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.

We have then implemented a more complex solution, in which we first cluster the available traces (resorting to the well-known K-Means algorithm [12]), and then select one representative case for each cluster (i.e. the cluster mean). Specifically, we 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 representative case).

An extensive experimental work is foreseen in the next months, in order to test the advantages of PBR with respect 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.

5 Incorporating the Tool in the ProM Framework

The ProM framework [25] is an open source environment, which offers a wide variety of process mining and analysis techniques. Process mining techniques [8]

allow for extracting information recorded in traces of actions. Traces can be mined to discover models describing processes, organizations, and products. Moreover, it is possible to use process mining to monitor deviations and exceptions from the underlying process schema. Unlike classical data mining techniques, the focus is on processes and on questions that transcend the simple performance-related queries supported by classical Business Intelligence commercial tools.

ProM is platform independent as it is implemented in Java. It is easy to add new plug-ins to ProM, without the need to recode parts of the system. Moreover, ProM allows for the import from and the export to a wide variety of formats and systems, and provides advanced visualization and verification capabilities.

We are currently working at an implementation of our retrieval and clustering facilities as a pair of ProM plug-ins. Our plug-ins will be used to support an analysis of deviations from the mined process schema. In particular, they 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 log), and performance analysis plug-ins (basic statistical analysis and performance analysis with a given process model), already embedded in ProM. Through such interactions, our work will globally support a principled reengineering activity, in line with the objectives described in the Introduction.

In order to be completely up-to-date, we are integrating our work with the latest version of ProM, namely ProM 6.

To start, we have structured the plug-ins architecture as required by ProM 6, i.e. as a set of three separate modules:

1. a data import module;
2. an analysis module;
3. a data visualization module.

Data import (1) was actually already provided in ProM 6 (while no import facility was available in the previous version ProM 5.2); our work is correctly interfaced with such a default instrument.

As for module (2), according to ProM 6 specifics, in our architecture it is further subdivided into: (2-i) a graphical interface for parameter setting; and (2-ii) a core analysis module, which is devoted to perform retrieval (clustering, respectively), on the basis of the parameter values acquired through module (2-i).

As regards modules (2-i) and (3), in line with the advanced visualization capabilities implemented in ProM, we are realizing usable and user-friendly graphical interfaces.

As an example, figure 4 shows a snapshot of the retrieval parameter setting interface (module (2-i)).

On the other hand, figure 2 shows a snapshot of our clustering output (module (3)).

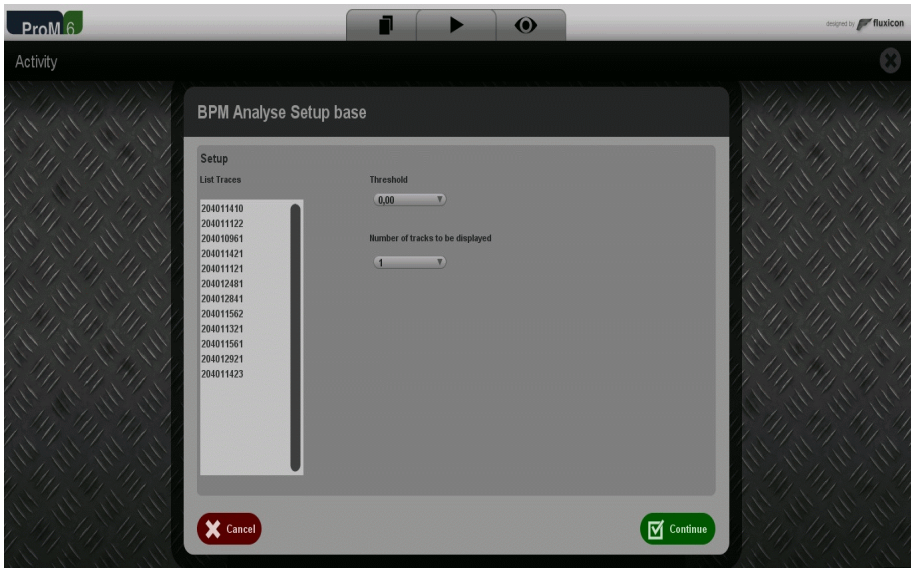

Fig. 4. A snapshot of retrieval parameter setting as rendered by the system graphical interface

With respect to version 5.2, ProM 6 is a little less consolidated, and less documentation is available. For this reason, our integration is requiring more time than what we originally foresaw; actually, the work is still in progress. In particular, our main ongoing effort is devoted to studying the requirements of the OpenXES library, the reference implementation of the XES standard for storing and managing event log data (http://www.xes-standard.org/openxes/start). As a matter of fact, we aim at being completely compliant with such requirements in traces management and elaboration (module (2)). Moreover, we are working at adapting our graphical modules (modules (3) and (2-i)), in order to make them correctly invokable within the ProM 6 environment. We plan to complete the implementation of our ProM 6 plug-ins in the near future.

A view of how our framework is meant to be used is reported in figure 5. Namely, as explained in the Introduction, end users and process engineers can exploit the tool in order to retrieve/cluster health care data. Clustering results, in particular, can highlight frequent anomalies, which could lead process engineers to identify problems in patient management, to be formally verified through other ProM functionalities. Moreover, process engineers may want to define a new version of the process schema, able to incorporate frequent changes. The data produced by following to the new schema will then be collected and analyzed as well, in a cyclical fashion.

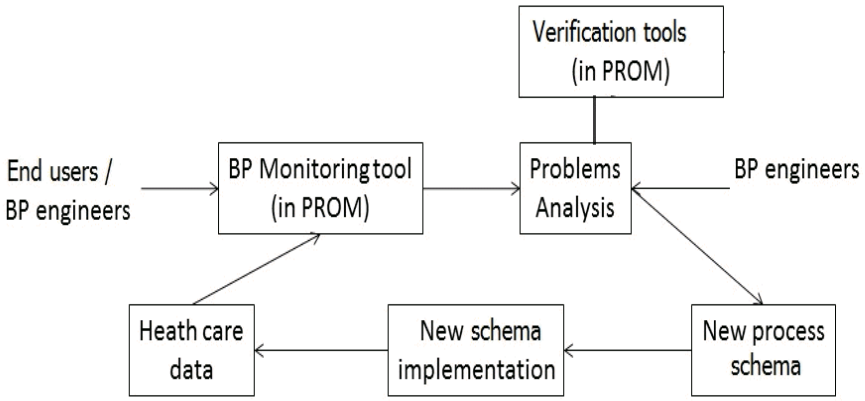

Fig. 5. A view of how the framework is meant to be exploited by end users and process engineers

6 Discussion and Conclusions

In this paper, we have described a case retrieval and clustering approach to BP monitoring, and we have introduced some recent research directions we are following, in order to improve our work.

Our main methodological contribution consists in the definition of a proper distance function, which is then resorted to both for retrieval and for clustering purposes, thus supporting end users as well as process engineers in their tasks.

In the literature a number of distance measure definitions for process instances exist. However, these definitions typically require further information in addition to the workflow structure, such as semantic annotations [26], or conversational knowledge. 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 [6] 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. 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. 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 properly compare action durations, and qualitative and quantitative constraints between actions. 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 [11]. 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 [11] 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. Finally, the distance function in [11] does not exploit action duration, and does not rely on taxonomical information about actions, as we do. Moreover, the work in [11] does not deal with (partially) overlapping actions in the sequence: only *before* and *meets* Allen's operators [3] are treated. We thus believe that our approach is more general and potentially more flexible in practice.

Another contribution [7] 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 relations between matched pairs of actions, resorting to the neighbors-graph distance [10], as we do. However, in [7] the alignment problem is strongly simplified, as they only match actions with the same name. Our approach is thus an extension to their work.

It is also worth citing the approach in [5], which adapts edit distance to trace clustering, 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.

As for our recent research lines, it is worth noting that the issue of avoiding exhaustive search in business process retrieval is quite a new research direction. An interesting paper addressing this topic is [4]. However, the approach followed in [4] is different from ours. First, in [4] the authors deal with graphs to represent their cases, while we resort on traces, that have a simpler structure. Second, the optimized search they propose, based on the A* algorithm, works on a partial mapping between the query case and the retrieved cases, where not all actions have been considered yet. On the other hand, the optimal (global) alignment between traces is automatically provided by our distance measure [16, 18]. Thus, their approach would not be directly applicable to our framework.

Our proposal to non-exhaustive search relies on a pivoting-based retrieval method. An extensive experimental work is foreseen in the next months, in order to test the advantages of PBR. In particular, 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.

Remarkably, our tool is also being incorporated in the ProM framework. The incorporation in ProM will allow our work to be used as a support to a principled reengineering activity, and will allow us to access plenty of new data, in order to complete our experimental work, in different application domains. Moreover, it will make the facility available to the BP management scientific community, thus enhancing the visibility of our CBR work.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations and systems approaches. AI Communications 7, 39–59 (1994)
2. Adams, H.P., Adams, R.J., Brott, T., del Zoppo, G.J., Furlan, A., Goldstein, L.B., Grubb, R.L., Higashida, R., Kidwell, C., Kwiatkowski, T.G., Marlerand, J.R., Hademenos, G.J.: Guidelines for the early management of patients with ischemic stroke. Stroke 34, 1056–1083 (2003)
3. Allen, J.F.: Towards a general theory of action and time. Artificial Intelligence 23, 123–154 (1984)
4. Bergmann, R., Gil, Y.: Retrieval of Semantic Workflows with Knowledge Intensive Similarity Measures. In: Ram, A., Wiratunga, N. (eds.) ICCBR 2011. LNCS, vol. 6880, pp. 17–31. Springer, Heidelberg (2011)
5. Jagadeesh Chandra Bose, R.P., Van der Aalst, W.: Context aware trace clustering: towards improving process mining results. In: Proc. SIAM Int. Conference on Data Mining, pp. 401–412. Springer (2000)
6. Bunke, H., Messmer, B.T.: Similarity Measures for Structured Representations. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) EWCBR 1993. LNCS, vol. 837, pp. 106–118. Springer, Heidelberg (1994)
7. Combi, C., Gozzi, M., Oliboni, B., Juarez, J.M., Marin, R.: Temporal similarity measures for querying clinical workflows. Artificial Intelligence in Medicine 46, 37–54 (2009)
8. Van der Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., Weijters, A.: Workflow mining: a survey of issues and approaches. Data and Knowledge Engineering 47, 237–267 (2003)
9. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. Wiley-Interscience, New York (2001)
10. Freska, C.: Temporal reasoning based on semi-intervals. Artificial Intelligence 54, 199–227 (1992)
11. Kapetanakis, S., Petridis, M., Knight, B., Ma, J., Bacon, L.: A Case Based Reasoning Approach for the Monitoring of Business Workflows. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 390–405. Springer, Heidelberg (2010)
12. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press (1967)
13. Madhusudan, T., Zhao, J.L., Marshall, B.: A case-based reasoning framework for workflow model management. Data and Knowledge Engineering 50, 87–115 (2004)

14. Minor, M., Tartakovski, A., Schmalen, D., Bergmann, R.: Agile workflow technology and case-based change reuse for long-term processes. International Journal of Intelligent Information Technologies 4(1), 80–98 (2008)
15. Montani, S.: Prototype-based management of business process exception cases. Applied Intelligence (published online in February 2009), doi: 10.1007/s10489-009-0165-z
16. Montani, S., Leonardi, G.: A Case-Based Approach to Business Process Monitoring. In: Bramer, M. (ed.) IFIP AI 2010. IFIP AICT, vol. 331, pp. 101–110. Springer, Heidelberg (2010)
17. Montani, S., Leonardi, G.: Trace retrieval and clustering for business process monitoring. Technical Report TR-INF-2012-03-01-UNIPMN, University of Piemonte Orientale (2012) (submitted to Information Systems journal)
18. Montani, S., Leonardi, G., LoVetere, M.: 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)
19. Palmer, M., Wu, Z.: Verb Semantics for English-Chinese Translation. Machine Translation 10, 59–92 (1995)
20. Portinale, L., Torasso, P., Magro, D.: Selecting Most Adaptable Diagnostic Solutions Through Pivoting-Based Retrieval. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 393–402. Springer, Heidelberg (1997)
21. Saposnik, G., Black, S.E., Hakim, A., Jiming, F., Tu, J.V., Kapral, M.K.: Age disparities in stroke quality of care and delivery of health services. Stroke 40, 3328–3335 (2009)
22. Sharan, R., Shamir, R.: CLICK: A clustering algorithm for gene expression analysis. In: Proc. International Conference on Intelligent Systems for Molecular Biology, pp. 260–268 (2000)
23. Socorro, R., Mico, L., Oncina, J.: A fast pivot-based indexing algorithm for metric spaces. Pattern Recognition Letters 32, 1511–1516 (2011)
24. Sokal, R., Michener, C.: A statistical method for evaluating systematic relationships. University of Kansas Science Bulletin 38, 1409–1438 (1958)
25. van Dongen, B., Alves De Medeiros, A., Verbeek, H., Weijters, A., Van der Aalst, W.: The proM framework: a new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) Knowledge Mangement and its Integrative Elements, pp. 444–454. Springer, Heidelberg (2005)
26. van Elst, L., Aschoff, F.R., Berbardi, A., Maus, H., Schwarz, S.: 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, pp. 340–345. IEEE Computer Society, Los Alamitos (2003)
27. Weber, B., Wild, W.: Towards the Agile Management of Business Processes. In: Althoff, K.-D., Dengel, A.R., Bergmann, R., Nick, M., Roth-Berghofer, T.R. (eds.) WM 2005. LNCS (LNAI), vol. 3782, pp. 409–419. Springer, Heidelberg (2005)
28. Yip, A.M., Chan, T.F., Mathew, T.P.: 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)