ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-639

A Cluster-Based Approach to Improve Similarity-Based Retrieval for Process-Oriented Case-Based Reasoning

Gilbert Müller and **Ralph Bergmann**[1]

Abstract. In case-based reasoning, improving the performance of the retrieval phase is still an important research issue for complex case representations and computationally expensive similarity measures. This holds particularly for the of retrieval workflows, which is a recent topic in process-oriented case-based reasoning. While most index-based retrieval methods are restricted to attribute-value representations, the application of a MAC/FAC retrieval approach introduces significant additional domain-specific development effort due to design the MAC phase. In this paper, we present a new index-based retrieval algorithm, which is applicable beyond attribute-value representations without introducing additional domain-specific development effort. It consists of a new clustering algorithm that constructs a cluster-based index structure based on case similarity, which helps finding the most similar cases more efficiently. The approach is developed and analyzed for the retrieval of semantic workflows. It significantly improves the retrieval time compared to a linear retriever, while maintaining a high retrieval quality. Further, it achieves a similar performance than the MAC/FAC retriever if the case base has a cluster structure, i.e., if it contains groups of similar cases.

1 INTRODUCTION

The retrieval phase in case-based reasoning (CBR) aims at selecting the k-most similar cases from a case-base for a given query considering a specific similarity measure. The overall goals of retrieval are to ensure retrieval completeness, i.e., to guarantee that the retrieval result does not miss any of the k-most similar cases and to ensure that the retrieval time is within an acceptable time limit. While linear retrieval algorithms, which compute the similarity between the query and each case of the case base, can easily ensure completeness, the retrieval speed is insufficient if the case base is too large, the case representation is too complex, or the similarity measure is computationally expensive. Thus, research in CBR has lead to several retrieval algorithms that improve retrieval speed without sacrificing completeness significantly. For cases represented as attribute-value pairs, various efficient index-based methods exist, such as case-retrieval nets [10] or kd-trees [17]. However, improving the retrieval speed is still an important research issue for more complex case representations. This is particularly true for process-oriented case-based reasoning (POCBR) in which a case represents a process or a workflow consisting of many task and data items linked together [2, 3, 9, 13, 14]. Such cases are usually represented as semantically labeled graphs and the similarity assessment requires a kind of inexact subgraph matching, which is computationally expensive. In such cases, existing index-based methods are not applicable. Recent research [9, 4] has addressed this problem by a two-phase retrieval, also called MAC/FAC ("Many are called, but few are chosen") retrieval [6]. The first retrieval phase (MAC phase) performs a rough pre-selection of a small subset of cases from a large case base. Then, the second phase (FAC phase) is executed to perform the computationally expensive graph-based similarity computation on the pre-selected cases only. This method improves the retrieval performance, if the MAC stage efficiently selects a small number of relevant cases. However, there is a risk that the MAC phase introduces retrieval errors, as it might disregard highly similar cases due to its limited assessment of the similarity. Hence, the retrieval approach for the MAC phase must be designed very carefully such that it is efficient and sufficiently precise in assessing the similarity. MAC/FAC improves retrieval performance but it introduces an additional significant development effort into a CBR system. To implement the MAC phase a second retrieval method must be developed. This requires defining an additional simplified domain specific case representation as well as a method for the pre-selection of cases, e.g., by an additional similarity measure applied to the simplified case representation. As the MAC phase must be aligned with the FAC phase, MAC/FAC not only increases the development effort but also the maintenance effort for a CBR application.

The aim of this paper is to develop a new index-based retrieval approach that allows to improve the retrieval performance of CBR applications without making assumptions concerning the similarity measure and the case representation, thus not increasing the development and maintenance effort. However, it will be illustrated by semantic workflows and a semantic workflow similarity [2]. This method significantly improves the state-of-the art in retrieval as it can be applied beyond pure attribute-value representations.

The basic idea behind the construction of the index is to use the similarity measure that is anyway modeled for retrieval to construct a hierarchical cluster-tree. This cluster-tree partitions the case base into sets of similar cases and it is used as an index structure during retrieval. Traversing the cluster tree allows finding clusters with cases similar to the query, thus reducing the number of required similarity computations. Unlike most existing cluster-based retrieval methods, which are restricted to attribute-value representations [5] or textual representations [7], our approach is applicable in the field of POCBR.

The next section introduces our previous work in POCRB, including semantic workflow representation and similarity. Then, we present a hierarchical clustering algorithm HBPAM to build a cluster-based index structure. Section 4 describes the cluster-based retrieval algorithm QTD. The experimental evaluation of retrieval time and quality based on a case base of cooking workflows is described in section 5. Finally, the paper discusses the results and presents prospective future work.

[1] University of Trier, Germany, email: [muellerg][bergmann]@uni-trier.de

2 PROCESS-ORIENTED CBR

Important goals of POCBR [13] are the creation of new workflows by reuse of best-practice workflows from a repository [2, 9] and the monitoring of running workflows [14]. These are important problems in traditional workflow areas as well as in new application domains such as e-science, e-health, or how-to knowledge from the web. POCBR supports the retrieval of workflow cases [9, 2, 4] and may in addition support their adaptation [12]. It requires an appropriate case representation for workflows as well as a similarity measure that allows to assess the utility of workflow for a new purpose. We now briefly describe our previous work, which is illustrated by an example from the domain of cooking [16].

2.1 Representation of Semantic Workflows

Broadly speaking, workflows consist of a set of *activities* (also called *tasks*) combined with *control-flow structures* like sequences, parallel (AND split/join) or alternative (XOR split/join) branches, and loops. Tasks and control-flow structures form the *control-flow*. In addition, tasks exchange certain *products*, which can be of physical matter (such as ingredients for cooking tasks) or data. Tasks, products, and relationships between the two of them form the *data flow*. Today, graph representations for workflows are widely used. In our work, we use a workflow representation based on semantically labeled graphs [2]. We represent a workflow as a directed graph $W = (N, E, S, T)$ where N is a set of nodes and $E \subseteq N \times N$ is a set of edges. Nodes and edges have types (e.g. data node, task node, see Fig. 1) assigned by the function T. Further, nodes and edges have semantic description from a language Σ, which is assigned by the function S. Σ is a semantic meta data language that is used for the semantic annotation of nodes and edges. We represent semantic descriptions in an object-oriented fashion to allow the application of well-established similarity measures from case-based reasoning [1]. Figure 1 shows a simple fragment of a workflow graph from the cooking domain with different types of nodes and edges. The graph for a workflow has one workflow node. The task nodes and data nodes represent tasks and data items, respectively. The data-flow edge is used to describe the linking of the data items consumed and produced by the tasks. The control-flow edge is used to represent the control-flow of the workflow, i.e., it links tasks with successor tasks or control-flow elements. For some nodes semantic descriptions are sketched, specifying ingredients used (data nodes) and tasks performed (cooking steps). The semantic descriptions are based on an ontology of data items and tasks, i.e., an ontology of ingredients and cooking steps.

2.2 Semantic Similarity

Our framework for modeling semantic workflow similarity extends traditional approaches for similarity in CBR and allows to model similarity measures which are inline with experts assessments [2]. The core of the similarity model is a local similarity measure for semantic descriptions $sim_\Sigma : \Sigma^2 \rightarrow [0, 1]$. In our example domain the taxonomical structure of the data and task ontology is employed to derive a similarity value that reflects the closeness in the ontology. It is combined with additional similarity measures that consider relevant attributes, such as the quantity of an ingredient used in a recipe (see [2] for more details and examples). The similarity $sim_N : N^2 \rightarrow [0, 1]$ of two nodes and two edges $sim_E : E^2 \rightarrow [0, 1]$ is then defined based on sim_Σ applied to their assigned semantic descriptions. The similarity $sim(QW, CW)$ between a query workflow QW and a case workflow CW is defined by means of an admissible mapping $m : N_q \cup E_q \rightarrow N_c \cup E_c$, which is a type-preserving, partial, injective mapping function of the nodes and edges of QW to those of CW. For each query node and edge x mapped by m, the similarity to the respective case node or edge $m(x)$ is computed by $sim_N(x, m(x))$ and $sim_E(x, m(x))$, respectively. The overall workflow similarity with respect to a mapping m, named $sim_m(QW, CW)$ is computed by an aggregation function (e.g. a weighted average) combining the previously computed similarity values. The overall workflow similarity is determined by the best possible mapping m, i.e.,

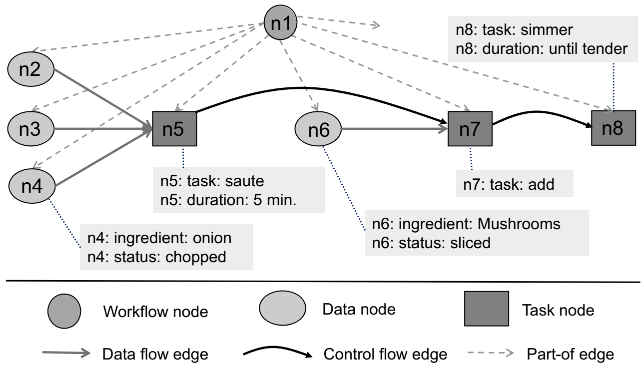

Figure 1. A sample workflow graph

$$sim(QW, CW) = \max\{sim_m(QW, CW) \mid \text{admissible map}\, m\}.$$

This similarity measure assesses how well the query workflow is covered by the case workflow. In particular, the similarity is 1 if the query workflow is exactly included in the case workflow as a subgraph. Hence, this similarity measure is not symmetrical.

2.3 Similarity Computation and Case Retrieval

The computation of the similarity requires the systematic construction of admissible mappings m. As the similarity computation by exhaustive search is computationally not feasible, we developed several memory-bounded A* search algorithms for this purpose [2]. For linear retrieval, this similarity computation must be applied to each case of the case base to find the k-most similar cases. By interleaving the A* search processes that occur for each case of the case base, we achieved an improved variant of a linear retriever, which we call *A* parallel retriever*. It is complete as long as the memory bounds are not exceeded but it is not sufficiently fast.

Recently, we also introduced a MAC/FAC approach [4] for workflow retrieval. Unlike the A* parallel retriever, the MAC/FAC retriever is not complete. It allows to speed-up the retrieval by the cost of retrieval errors. Thus, there is a trade-off between retrieval speed and retrieval quality, which can be controlled by a parameter of the MAC phase. The MAC/FAC approach uses a MAC phase with a domain specific attribute-value representation as well as a similarity measure with appropriately weighted local similarity measures for the attributes. The additional modeling effort introduced thereby is a significant disadvantage of the MAC/FAC approach and the motivation for research presented in this paper.

3 CLUSTER INDEX CONSTRUCTION

We now introduce the *Hierarchical Bisecting Partitioning Around Medoids* (HBPAM) algorithm, which is a hierarchical version of the traditional *Partitioning Around Medoids* (PAM) algorithm [8]. It combines a bisecting PAM approach [11] with a hierarchical k-means approach [15] in order to construct a hierarchical index where each cluster is represented by a particular case (medoid). Such an index can currently not be gained using traditional clustering algorithms. HBPAM constructs a binary cluster tree for a given case base CB as follows:

```
INPUT: case base CB, number of runs I
OUTPUT: cluster tree T
BEGIN
 Initialize the tree T with a root cluster RC
   consisting of all cases in CB
 REPEAT
  Select a leaf cluster C from T with |C|>=2
  Execute PAM I times to split C into two
    sub-clusters C1 and C2
  Select PAM result (C1,C2) with best quality
  Link C1 and C2 as child clusters to C in T
 UNTIL each leaf cluster contains one case
 RETURN T
END
```

Initially, HBPAM assigns all cases of the case base to a root cluster RC. Using PAM, HBPAM repeatedly splits the cases of a cluster C into two sub-clusters and thereby constructs the cluster tree T in a top-down fashion (see Fig. 2). As PAM only finds a local optimum of the clustering depending on the randomly selected initial medoids, multiple runs (parameter I) are executed to alleviate this problem [15]. The best result is selected based on the quality criterion of PAM, which maximizes the average similarity of the medoids to the cases within the clusters. The two resulting clusters C1 and C2 are linked as *child clusters* of C. When the algorithm terminates, each *leaf cluster* (depicted as rectangle in Fig. 2), contains only one case.

As our case base consists of workflows, the similarity measure used by PAM to assign objects to medoids and to compute the clustering quality is based on the semantic similarity measure (section 2.2) used for retrieval. Because this similarity measures is asymmetric, we apply the min-symmetrization method [3] during clustering, i.e., $min\{sim(W_1, W_2), sim(W_2, W_1)\}$ is the similarity between the workflows W_1 and W_2. To avoid multiple computations of the similarity between the same workflow pairs, the similarity values computed once are cached.

Since PAM, which has a complexity of $O(l(n-l)^2)$, is executed I times for $n = |CB|$ cases and with $l = 2$ clusters, the complexity of the computation in the inner loop is $O(I \cdot 2(n-2)^2) = O(I \cdot n^2)$. In the worst case, the HBPAM algorithm produces a most unbalanced tree, which requires to execute the split operation n times. This results in an overall complexity of $O(|CB|^3)$.

4 CLUSTER-BASED RETRIEVAL

Based on the HBPAM cluster index, we developed a hierarchical retrieval algorithm, named *Queued Top-Down Cluster-Based Retrieval* (QTD) for retrieving the k-most similar cases w.r.t. a query workflow QW. The basic idea is to identify some 'reasonably-sized' clusters from the tree that are similar to the query and to investigate the cases

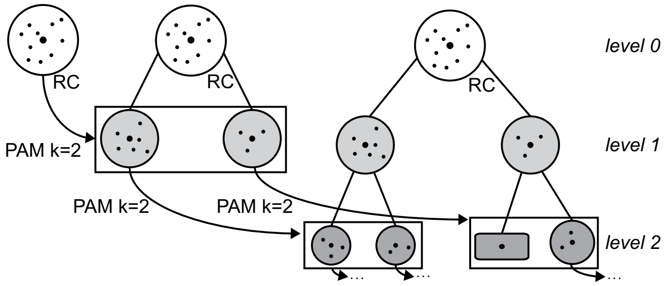

Figure 2. HBPAM example

of those clusters for similarity only. To control the search for clusters, QTD uses two parameters *upper level* UL and *lower level* LL which define two levels[2] in cluster tree. The search is restricted to the interval of nodes between these two levels. Additionally, a parameter filter size $FS \leq k$ defines the minimum total number of cases from the most similar clusters that have to be investigated before the retrieval stops. In the algorithm, the similarity between a cluster C and a query QW is determined based on the medoid of the cluster, i.e., $sim(QW, C) = sim(QW, medoid(C))$.

```
INPUT: query QW, cluster tree T,
       upper level UL, lower level LL,
       filter size FS, number of cases k
OUTPUT: list of k-most similar cases RL
BEGIN
   Initialize sorted queue SQ, result list RL
      cluster retrieval list CRL
   FOR cluster C FROM clustersAtLevel(UL)
      SQ.PUSH(C,sim(QW,medoid(C)))
   END FOR
   REPEAT
      X=SQ.POP()
      IF (size(X)!= 1 AND level(X)!=LL) THEN
         simL=sim(QW,medoid(childLeft(X)))
         simR=sim(QW,medoid(childRight(X)))
         SQ.PUSH(childLeft(X),simL)
         SQ.PUSH(childRight(X),simR)
      ELSE
         CRL.add(cases(X))
      END IF
   UNTIL size(CRL)>=FS
   RL = EXECUTE A*Parallel(QW,CRL,k)
   RETURN RL
END
```

The QTD retriever maintains a queue SQ that stores a cluster-similarity-pair for each computed similarity between a query QW and a cluster C, i.e., $(C, sim(QW, C))$. At any time QW is in descending order w.r.t. the similarity values. Initially, the similarity of the query QW to each cluster $C \in T$ at level UL is calculated and their cluster-similarity-pairs are added to SQ. The following iteration implements a heuristic search in the tree from the nodes of level UL towards some nodes of level LL which are most similar to the query. Therefore, the first cluster of SQ (cluster with highest similarity) is selected and removed from SQ. If this cluster is not a leaf

[2] The level of a node is the number of parent nodes the node has. The root node RC is at level 0.

node or on the lowest level LL, the similarities of the query QW to the left and right child cluster are computed and both child clusters are added along with their similarity into SQ. Otherwise, the cases contained in the cluster are added to the cluster retrieval list CRL. This iteration is continued until the CRL contains at least FS many cases. Then, for each of the cases in CRL, the similarity w.r.t. the query QW is computed using a linear retriever, to determine the k-most similar cases which are stored in the result list RL. In our case we use the A* parallel retriever for semantic workflows (see Sec. 2.3).

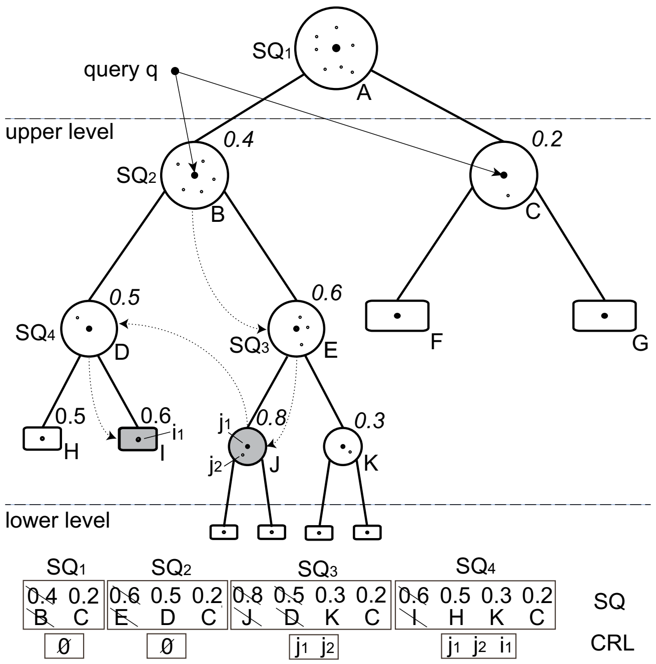

Figure 3. QTD example ($UL = 1, LL = 3, FS = 3, k = 1$)

An example scenario is illustrated in Figure 3. First, the cluster-similarity-pairs for each cluster at level $UL = 1$ (B and C) are added to SQ. Next, cluster B, the most similar cluster in SQ, is removed from SQ. Then, the cluster-similarity-pairs of the child clusters of B (D and E) are inserted into SQ. Next, this process is again applied to cluster E. As the subsequent cluster J is located at level LL its cases j_1 and j_2 are added to the cluster retrieval list CRL and no further clusters are added to SQ. Next, cluster D is selected and its child clusters are inserted to SQ. Finally, leaf cluster I is selected and the case i_1 of I is added to CRL. Now the iteration stops as the stopping criterion $size(CRL) >= 3$ is met. The similarity between the query and the three cases in CRL is computed by applying the A* parallel retriever and the most similar case is returned (as k=1).

Please note that this retrieval algorithm does not make any assumption concerning the similarity measure used during retrieval. HBPAM and QTD just use the available similarity measure for clustering and retrieval. Also, no assumptions are made concerning the structure of the cases. Hence, this method is generic and cannot only be applied for workflow cases.

The QTD retriever can reduce the retrieval time compared to a linear retriever, as the number of overall similarity computations can be reduced, depending on the chosen parameter values. The overall number of similarity computations performed is the number of computations performed during tree traversal (one similarity computation for each cluster being investigated) plus the size of the resulting CRL, as for each case in CRL the similarity is computed as well. In the worst case, $2^{LL+1} - 2^{UL}$ similarity computations are performed during tree traversal, which occurs if each cluster is considered. In the best case, the tree is traversed top-down and the algorithm terminates when selecting the first cluster at level LL, hence $2^{UL} + 2 \cdot (LL - UL)$ similarity computations have to be performed. The size of the CRL can be estimated by $FS + \mathcal{E}$, whereas $\mathcal{E} < \max\{size(C)|\text{Clusters C at level LL}\}$. Thus, in the best case with a balanced tree and with $UL = 1$, $FS = k$, and $LL = \lfloor log_2(|CB|/k) \rfloor$ the tree can be traversed from the root node to a node that contains the requested number of cases, leading to $2 \cdot LL + k$ similarity computations, which would be a significant search reduction.

An other aspect is also important to note: the QTD retriever does not guarantee retrieval completeness. The selected clustering strategy cannot guarantee that the selected clusters contain the k-most similar cases. Hence it could happen that similar cases are missed because they are in a different cluster. The larger the filter size parameter FS, the more similar clusters are investigated and thereby the chance of missing cases is reduced. Thus, there is a trade-off between retrieval quality and retrieval speed, similar to MAC/FAC approaches. The retrieval time and retrieval error (or quality) obviously depends on many factors, including the distribution of the cases, i.e., how well the cases can be grouped into clusters of similar cases. This makes a more thorough theoretical assessment difficult. Hence, we investigate the characteristics of the retrieval algorithm using an experimental approach.

5 EXPERIMENTAL EVALUATION

We now evaluate retrieval time and retrieval quality for various parameter combinations of the QTD algorithm.

While measuring retrieval time is obvious, different measures for retrieval quality have been proposed in the literature. We use a measure that assesses which cases from the set of the k-most similar cases (called $MSC(QW, k)$ in equation 1) have been omitted in the result list RL returned by QTD or MAC/FAC retrieval for the query QW. Each missing case has a negative impact on the retrieval quality proportional to its similarity to the query. Thus, if a highly similar case is omitted, the negative impact on the quality is stronger than if a case with a low similarity is omitted.

$$quality(QW, RL) = 1 - \frac{1}{|RL|} \cdot \sum_{CW \in MSC(QW,|RL|)-RL} sim(QW, CW) \quad (1)$$

In our experiments, we investigated five hypotheses. We explore whether the two level parameters (Hypothesis $H1$) as well as the filter size parameter (Hypothesis $H2$) can be used to control the trade-off between retrieval quality and retrieval time. Moreover, we expect that the distribution of the cases has an impact on the retrieval quality. We assume that a case base with a strong cluster structure, i.e., a case base in which there are separated groups of similar cases leads to a higher retrieval quality compared to a case base with litte cluster structure, i.e., where cases are more equally distributed (Hypothesis $H3$). Further, we compare the retrieval time of QTD with the retrieval time of the A* parallel retriever. We expect that QTD decreases the retrieval time of the A* parallel retriever with an acceptable loss of retrieval quality (Hypothesis $H4$). Finally, we compare QTD with our MAC/FAC retriever [4]. We do not expect to improve retrieval time and quality compared to MAC/FAC (Hypothesis $H5$) as QTD is generic, while MAC/FAC uses additional, manually optimized domain knowledge in the MAC phase.

Table 1. Evaluation parameters

parameter	values
number of cases k	10, 25, 50, 100, 200
filter size FS	10, 25, 50, 100, 200, 300
upper level UL	1, 2, ..., 12
lower level LL	7, 8, ..., 12

We implemented the QTD algorithm as new component within the CAKE framework[3] in which the A* parallel retriever and the MAC/FAC retriever were already included. This allows a systematic comparison of the three retrieval algorithms in the same implementation environment. The evaluation was performed in cooking domain using a workflow repository automatically generated by information extraction [16] from the recipe web site allrecipes.com. Furthermore, we used a manually developed cooking ontology with 208 ingredients and 225 cooking preparations steps upon on which the semantic similarity measure has been defined. The workflow repository consist of 1526 cases (case base CB-I). A first cluster analysis of this case base revealed that it has only little cluster [3] structure. Hence, to assess $H3$, a second case base (CB-II) with a strong cluster structure is constructed, which consists of 1793 cases. For this purpose, we randomly selected 50 cases from CB-I and generated 35-40 variations of each case by randomly modifying task orders, and by adding, deleting, or replacing nodes and edges. Both case bases are queried using a set of 100 query cases (also recipes extracted from allrecipes.com), which are different from the cases in the two case bases. QTD was executed with all parameter combinations given in table 1 and HB-PAM with parameter $I = 4$.

Table 2. Evaluation of the level parameters

		CB-I		CB-II	
UL	LL	quality	time	quality	time
1	7	0.6968	0.4925	0.7256	0.5298
2	7	0.6963	0.4996	0.7575	0.5756
4	7	0.7378	0.5276	0.8172	0.6005
6	7	0.7644	0.6220	0.8571	0.6706
7	7	0.7787	0.7614	0.8855	0.7963
1	8	0.7067	0.5487	0.7345	0.6005
2	8	0.7053	0.5363	0.7702	0.6418
4	8	0.7460	0.5868	0.8269	0.6973
6	8	0.7706	0.6870	0.8649	0.7663
7	8	0.7803	0.8191	0.8855	0.8801
8	8	0.8003	1.0577	0.9097	1.1063
1	9	0.7163	0.6213	0.7437	0.6868
2	9	0.7160	0.6118	0.7796	0.7415
4	9	0.7548	0.6748	0.8316	0.8038
6	9	0.7786	0.7735	0.8692	0.8757
7	9	0.7866	0.9032	0.8876	0.9921
8	9	0.8009	1.1299	0.9084	1.2028
9	9	0.8197	1.6250	0.9270	1.4883

Additionally the A* parallel retriever and the MAC/FAC retriever were executed with the same queries on CB-I and CB-II. All experiments were executed on a PC with an Intel Core i7-870 CPU @ 2.93 GHz and 8 GB RAM running Windows-7 Enterprise 64-bit.

[3] Collaborative Agile Knowledge Engine, see cake.wi2.uni-trier.de

First, we evaluated the impact of level parameters on the trade-off between retrieval time and quality (see $H1$). Table 2 shows an extract of different upper and lower level combinations and their average time and quality values over all parameter combinations in Tab. 1.

For both level parameters UL and LL it can be observed that a larger value leads to a higher quality but also to a higher retrieval time, which confirms $H1$. The impact of UL is higher than the impact of LL. Larger level parameter values lead to higher quality as the considered clusters are smaller and thus more cases from different clusters are collected in the CRL. The fastest retrieval is achieved when $UL = 1$, which is an indication that the worst-case assumption considered in section 4 does not occur in the experiment. This shows that climbing down the cluster tree indeed reduces the candidate clusters to be investigated and reduces retrieval time. When comparing the results for CB-I and CB-II we can also see that the quality is much better for the case base with cluster structure, which is in line with hypothesis $H3$.

Table 3. Comparison of retrievers for different parameters k and FS

		QTD		MAC/FAC		A* Parallel
k	FS	quality	time	quality	time	time
		CB-I				
10	10	0.6042	0.2469	0.7537	0.2506	1.3280
10	25	0.6431	0.2790	0.8802	0.3005	1.3280
10	50	0.7026	0.3240	0.9563	0.3582	1.3280
10	100	0.7664	0.3985	0.9829	0.4423	1.3280
25	25	0.6148	0.3147	0.7892	0.3103	1.4370
25	50	0.6707	0.3717	0.9040	0.3937	1.4370
25	100	0.7402	0.4595	0.9620	0.5176	1.4370
50	50	0.6521	0.4188	0.8278	0.4272	1.5640
50	100	0.7227	0.5211	0.9286	0.5985	1.5640
		CB-II				
10	10	0.7201	0.2596	0.7182	0.2889	1.3221
10	25	0.7796	0.3031	0.8503	0.3355	1.3221
10	50	0.8228	0.3606	0.9219	0.4117	1.3221
10	100	0.8660	0.4660	0.9586	0.5276	1.3221
25	25	0.7543	0.3239	0.7865	0.3365	1.4407
25	50	0.8042	0.3863	0.8838	0.4348	1.4407
25	100	0.8542	0.5085	0.9490	0.5723	1.4407
50	50	0.7806	0.4257	0.8199	0.4624	1.6215
50	100	0.8478	0.5577	0.9184	0.6114	1.6215

For the following analyses, we fixed the level parameters to $UL = 6$ and $LL = 7$ as this setting is a good compromise between retrieval speed and quality. First, we evaluated the influence of filter size FS and number of cases k to be retrieved. Table 3 shows the average quality and retrieval time measures over all 100 queries.

As expected, the retrieval time increases both with the increase of k and FS as both parameters increase the number of similarity assessments in QTD. Further, for a fixed number of cases k, an increase of FS increases the quality, as more cases are investigated, which confirms hypothesis $H2$. Again, we can see that the quality results for CB-II are better than for CB-I, which is in line with hypothesis $H3$. Table 3 also allows to compare the three retrieval algorithms. Compared to A* parallel, QTD shows a speed-up in retrieval of a factor 3-5.4 for case base CB-I and a factor of 2.8-5.1 for CB-II. If we consider a quality value of 75% as acceptable, QTD achieves a retrieval speed-up of a factor 3.8-4.3 with a quality above this level. However, for CB-I the 75% level is only reached for $k = 10$ and $FS = 100$, but still leading to a speedup of a factor 3.3. Hence, hypothesis $H4$ is confirmed for CB-II, for CB-I the speed-up is significant, but the retrieval quality is becoming worse, which is in line with hypothesis $H3$.

When investigating the results for the MAC/FAC retriever, one can easily recognize that the retrieval quality does not depend on the cluster structure of the case base. Its quality on CB-I is clearly better than the quality of the QTD retriever, while the retrieval time is similar. However, for CB-II the difference in quality and retrieval time is getting smaller and thus the advantage of the MAC/FAC retriever disappears. Figure 4 additionally illustrates the relation between QTD and the MAC/FAC retriever. Here, the average retrieval time and quality values over the 100 queries are plotted, for $k = 10$ cases and a varying filter size of 10-100. Overall, hypothesis $H5$ is confirmed, as QTD does not outperform MAC/FAC, but for the case base CB-II the quality difference is surprisingly small. Hence, QTD achieves a similar performance than the MAC/FAC retriever on CB-II.

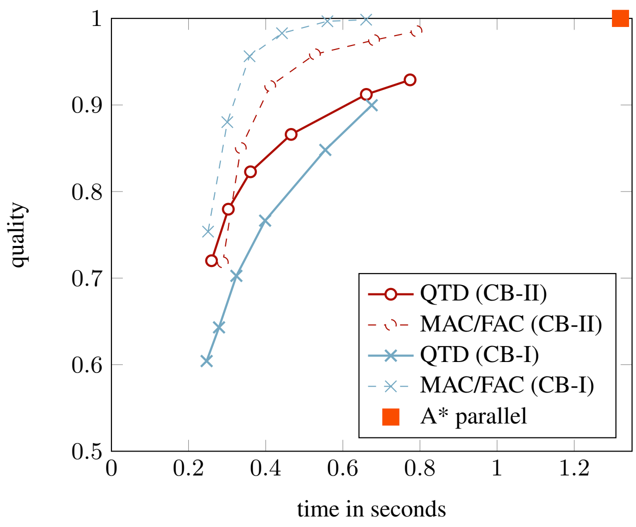

Figure 4. Retrieval time and quality for $k = 10$, $UL = 6$, $LL = 7$, and FS varying from 10-100.

6 CONCLUSIONS AND FUTURE WORK

We presented a new approach for index-based retrieval of workflows. A cluster index is constructed prior to the execution of a cluster-based retrieval. For this purpose we developed the HBPAM clustering algorithm and the QTD retrieval algorithm. Our investigation revealed that the presented approach is able to decrease the retrieval time without a considerable loss of retrieval quality compared to a linear retrieval approach. Furthermore, parameters enable to control the trade-off between retrieval quality and retrieval time. The retrieval quality of the presented approach depends on the structure of the case base; if groups of similar cases are present, our approach is able to compete with a MAC/FAC approach specified for semantic workflows. A significant advantage compared to the general MAC/FAC approach is that no additional retrieval phase (the MAC stage) must be designed and thus the development and maintenance effort is not increased.

In contrast to other indexing approaches, no restrictions are made on the similarity measure and the case representation. Furthermore, it solely relies on the similarity measure present in any CBR application. Hence, the approach could possibly serve as a generic index-based retrieval framework for CBR applications with different complex case representations and different similarity measures, which still has to be investigated.

Future work could also focus on a further improvement of the presented approach. Adapting the clustering quality criterion of HBPAM, for example, could improve the retrieval performance, since traditional clustering might not be optimal for the construction of an cluster index [18]. Further studies are needed to investigate whether this method is suitable for other case bases, other domains, different case representations and other similarity measures.

ACKNOWLEDGEMENTS

This work was funded by the German Research Foundation (DFG), project number BE 1373/3-1.

REFERENCES

[1] Ralph Bergmann, *Experience Management - Foundations, Development Methodology, and Internet-Based Applications*, volume LNAI 2432, Springer, 2002.
[2] Ralph Bergmann and Yolanda Gil, 'Similarity assessment and efficient retrieval of semantic workflows', *Inf. Syst.*, **40**, 115–127, (March 2014).
[3] Ralph Bergmann, Gilbert Müller, and Daniel Wittkowsky, 'Workflow clustering using semantic similarity measures.', in *KI 2013: Advances in Artificial Intelligence*, eds., Timm and Thimm, volume 8077 of *LNCS*, pp. 13–24. Springer, (2013).
[4] Ralph Bergmann and Alexander Stromer, 'Mac/fac retrieval of semantic workflows', in *Proc. of FLAIRS 2013, St. Pete Beach, Florida. May, 2013*, eds., Boonthum-Denecke and Youngblood. AAAI Press, (2013).
[5] Chuang-Cheng Chiu and Chieh-Yuan Tsai, 'A weighted feature c-means clustering algorithm for case indexing and retrieval in cased-based reasoning', in *New Trends in Applied Artificial Intelligence*, eds., Okuno and Ali, volume 4570 of *LNCS*, 541–551, Springer, (2007).
[6] Kenneth D. Forbus and Dedre Gentner, 'MAC/FAC: a model of similarity-based retrieval', in *Proc. of CogSci1991*. Cognitive Science Society, (1991).
[7] Nick Jardine and Cornelis Joost van Rijsbergen, 'The use of hierarchic clustering in information retrieval', *Information storage and retrieval*, **7**(5), 217–240, (1971).
[8] Leonard Kaufman and Peter J. Rousseeuw, *Finding Groups in Data - An Introduction to Cluster Analysis*, John Wiley, New York, 1990.
[9] Kendall-Morwick, J. and Leake, D., 'On tuning two-phase retrieval for structured cases', in *ICCBR-Workshop on Process-oriented Case-Based Reasoning*, pp. 25–34, Lyon, (2012).
[10] Mario Lenz and Hans-Dieter Burkhard, 'Lazy propagation in Case Retrieval Nets', in *Proc. of ECAI-96*, ed., Wahlster, pp. 127–131. John Wiley and Sons, (1996).
[11] Michael Steinbach, George Karypis and Vipin Kumar, 'A comparison of document clustering techniques', in *In KDD Workshop on Text Mining*, (2000).
[12] Mirjam Minor, Ralph Bergmann, and Sebastian Görg, 'Case-based adaptation of workflows', *Inf. Syst.*, **40**, 142–152, (2014).
[13] Mirjam Minor, Stefania Montani, and Juan A. Recio-García, 'Editorial: Process-oriented case-based reasoning', *Inf. Syst.*, **40**, 103–105, (2014).
[14] Stefania Montani and Giorgio Leonardi, 'Retrieval and clustering for supporting business process adjustment and analysis', *Information Systems*, **40**(0), 128 – 141, (2014).
[15] Rasha Kashef and Mohamed S. Kamel, 'Efficient bisecting k-medoids and its application in gene expression analysis', in *Proc. of ICIAR '08*, pp. 423–434, Berlin, Heidelberg, (2008). Springer-Verlag.
[16] Pol Schumacher, Mirjam Minor, Kirstin Walter, and Ralph Bergmann, 'Extraction of procedural knowledge from the web: A comparison of two workflow extraction approaches', in *Proc. of WWW '12*, pp. 739–747. ACM, (2012).
[17] Stefan Wess, Klaus-Dieter Althoff, and Guido Derwand, 'Using *kd*-trees to improve the retrieval step in case-based reasoning', in *Proc. of EWCBR-93*, eds., Wess, Althoff, and Richter, LNAI, 837, pp. 167–181. Springer, (1993).
[18] Tim Wylie, Michael A Schuh, John Sheppard, and Rafal A Angryk, 'Cluster analysis for optimal indexing', in *Proc. of FLAIRS 2013*, (2013).