Similarity Assessment and Efficient Retrieval of Semantic Workflows

Ralph Bergmann[a], Yolanda Gil[b]

[a] *University of Trier, Department of Business Information Systems II, Universitätsring, 54286 Trier, Germany*
[b] *Information Sciences Institute, University of Southern California, Marina del Rey, CA 90292, USA*

Abstract

In the recent years, the use of workflows has significantly expanded from its original domain of business processes towards new areas. The increasing demand for individual and more flexible workflows asks for new methods that support domain experts to create, monitor, and adapt workflows. The emergent field of process-oriented case-based reasoning addresses this problem by proposing methods for reasoning with workflows based on experience. New workflows can be constructed by reuse of already available similar workflows from a repository. Hence, methods for the similarity assessment of workflows and for the efficient retrieval of similar workflows from a repository are of core importance. To this end, we describe a new generic model for representing workflows as semantically labeled graphs, together with a related model for knowledge intensive similarity measures. Further, new algorithms for workflow similarity computation, based on A* search are described. A new retrieval algorithm is introduced that goes beyond traditional sequential retrieval for graphs, interweaving similarity computation with case selection. We describe the application of this model and several experimental evaluations of the algorithms in the domain of scientific workflows and in the domain of business workflows, thereby showing its broad applicability.

Keywords: Business Workflows, Scientific Workflows, Workflow Similarity, Retrieval, Case-Based Reasoning

1. Introduction

Business workflow management is an established area that aims at "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action,

Email addresses: bergmann@uni-trier.de (Ralph Bergmann), gil@isi.edu (Yolanda Gil)

according to a set of procedural rules" [29]. In the recent years, the use of workflows has significantly expanded from the original domain of business processes towards new areas, as the modeling capabilities and the execution support that workflows provide are widely applicable. For example, in e-science *scientific workflows* are executable descriptions of automatable scientific processes such as computational science simulations and data analyses [44]. In medical healthcare, workflows can be used to support the execution of *medical guidelines*, i.e. standardized treatment processes of a certain disease [35, 33]. Further, workflows can be used to represent and execute search [19] and information integration processes [24] in the context of decision support systems. Even in cookery, workflows can be used as a means to represent the cooking instructions within a recipe [36] in order to provide step-by-step guidance during cooking.

Such new applications of workflows typically deal with a number of new difficulties, particularly due to

- an increasing number of specific workflows potentially relevant to a domain,
- an increasing complexity of workflows,
- an increased demand for more flexibility, resulting in *agile workflows*, and
- the need to enable non-IT staff to create workflows and to control their execution.

These new challenges in workflow management ask for new methods and tools that support domain experts to perform workflow-related tasks, particularly workflow modeling, composition, adaptation, analysis, and optimization.

1.1. Process-oriented case-based reasoning

Case-based reasoning (CBR) [8, 6, 1] has recently demonstrated its high potential for this purpose [45, 34, 9, 30, 20, 16, 37]. The emergent field of process-oriented case-based reasoning (POCBR) particularly focusses on the integration of process-oriented information systems with case-based reasoning. The particular appeal of CBR comes from the fact that it supports and partially automates experience-based problem solving. Process-oriented case-based reasoning aims at addressing the problem of creating new workflows as an experience-based activity. To this end, new workflows can be constructed by reuse of already available workflows that have to be adapted for new purposes and circumstances. A repository of successful workflows reflecting best-practice in a domain is the core of a POCBR approach. In case-based reasoning terminology such a repository is called a *case-base*, i.e. a collection of cases, each of which describes a particular workflow addressing a particular goal. Users can query the repository with a specification of important properties of the workflow s/he wants to create in order to retrieve potentially reusable workflows. One particular characteristic of CBR is that it allows to find cases that do not match exactly the user's query, but which are at least similar in some respect. This is based on the core

assumption of CBR that similar problems have similar solutions. So, even if the repository does not contain a workflow which is immediately addressing the user's problem, a similar workflow could be available as a good starting point. Such similar workflows can then be adapted or at least provide some inspiration and guidance for the user during workflow construction.

1.2. Contributions of this paper

To implement such a POCBR approach, methods for the similarity assessment of workflows and for the efficient retrieval of similar workflows from a repository are of core importance. This paper addresses both problems in the following ways.

It is well known in CBR that the notion of similarity is crucial and that similarity must be modeled according to the particular domain at hand in order to ensure a high retrieval quality [8, 31, 6, 13]. In line with this observation, we propose a new general framework for workflow representation and related similarity modeling. Workflows are represented as semantically annotated graphs, extending previous proposals for graph-based workflow representations [38, 18, 37]. The well-known local/global principle to modeling similarity measures [13, 6] is extended to these graph representations, providing a flexible means for workflow similarity modeling.

Similarity-based retrieval in CBR is computationally difficult if the repository is getting large and if a domain-specific similarity measure should be used. Particularly for graph-based representations, the well-known indexing approaches [31] cannot be applied because they cannot cope with the graph structure. Even, the computational complexity of a single similarity assessment causes problems due to the fact that two graph structures must be compared. As a second contribution, this paper addresses this problem by developing several new algorithms for similarity computation and retrieval. With experimental implementations we analyzed the retrieval performance and quality of these algorithms. The most promising algorithm has then been implemented as a core component within the process-oriented CBR system CAKE [9, 38]. It interweaves similarity assessment and retrieval and thereby enables efficient retrieval even for larger repositories.

As we are interested in generic methods, we demonstrate our methods using two workflow domains with very different characteristics: we address traditional business processes, which are control-flow oriented and scientific workflows, which strongly focus on the data flow [32].

2. Foundations and related work

This section provides a focused introduction to the relevant foundations of (semantic) workflow representation and the notion of similarity in CBR. We also survey relevant related work in the area of workflow similarity and retrieval.

2.1. Workflow representation

Workflow representations typically reflect the dataflow and/or the control flow structure among a set of tasks of a process. That is, the workflow represents the partial ordering of task that are part of the overall process. Today, various workflow representation formats are used, depending on the kind of workflow. Representation approaches for business workflows have a strong focus on the control flow, usually implementing (some of) the workflow patterns proposed by van Aalst [17]. Typical control flow patterns are *sequence*, *and-split*, *and-join*, *xor-split*, *xor-join*, and possibly *loops*. Figure 1a shows an example of a business workflow within a University administration, according to the representation[1] used in the CAKE project [9, 38] at the University of Trier. The workflow describes a simple set of activities required to book a room in a university. First, one has to search for an available room in the online database. Then, one can either send a plain email with a booking request to the facility manager or alternatively one can fill in a room request form and send it. Then, the facility manager assigns the room and sends a confirmation. The graphical representation of this workflow just shows the six tasks involved, which are organized using sequences and one xor-split/join control-flow pattern. The flow of data, such as the filled form, is neglected in this representation.

Unlike business workflows, scientific workflows have a strong focus on the dataflow, typically restricting the control flow to a partial ordering of the tasks [32]. Such a simple control structure offers several advantages and has been sufficient to support a variety of applications [44]. Each task (also called component) in the scientific workflow can have input datasets and input parameters, as well as output datasets. Dataflow links indicate what data are output by a task and consumed by another task. Figure 1b shows an example of a scientific workflow describing some data mining process according to the representation in the Wings project [21]. This example uses three tasks taken from widely known Weka component library [46]. The library of workflow components includes a hierarchy of algorithm types that represent abstract classes of steps. For example, the *ID3*, *LMT*, and *J48* algorithms are placed in the hierarchy under the class of *Decision Tree Modelers*, which is itself a subclass of the class *Modeler* (see also Fig. 4). Workflow templates such as the one shown in Fig. 1b can include abstract classes or concrete algorithms. The shown example workflow first resamples the data set *InputData1*, then discretizes the result, and then applies an ID3 modeler to create a model. The graphical representation shows the tasks together with the exchanged data and the parameters (e.g. *numberOfBins*). The control flow is not explicitly shown but the necessary sequential order immediately results from the shown data flow.

2.2. Semantic workflows

Semantic workflow representations enrich the above described workflow formats by adding metadata and constraints to the individual elements of the

[1] The graphical representation was inspired by UML activity diagrams.

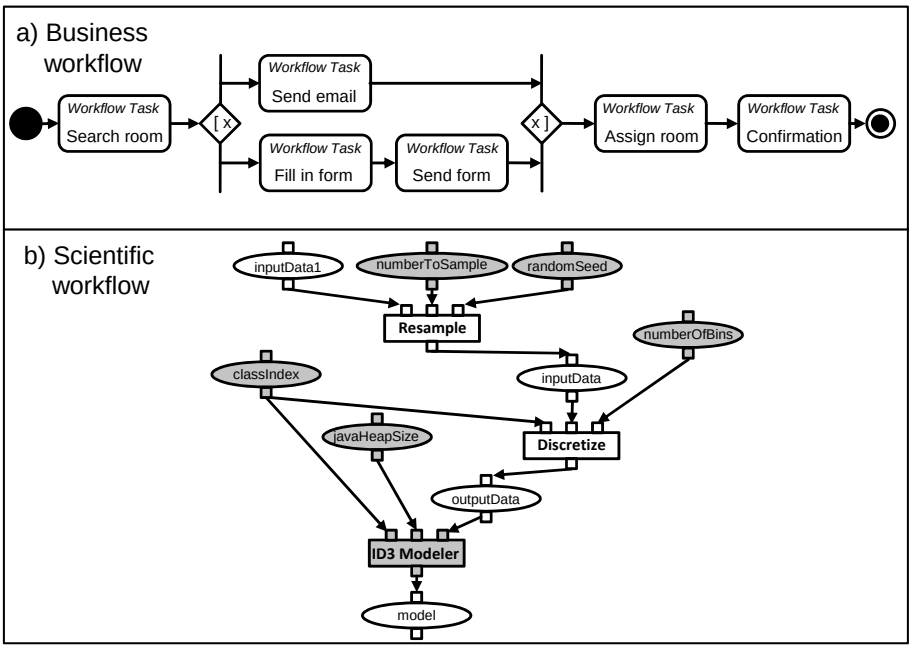

Figure 1: Two examples of different workflows.

workflow. Ontologies can be used to formalize the respective workflow domain by organizing the metadata descriptions of the occurring workflow elements. This allows to capture the semantics of tasks, data items, and control flow items. Semantic annotations of workflows are important as they can be used as basis for the similarity assessment (see Sect. 2.3). Semantic similarity measures (also called knowledge-intensive similarity measures) allow to define workflow items (e.g. tasks or data) as similar if they have a similar meaning. Hence, they are the foundation that allows us to derive semantic similarity measures for workflows as a whole.

The workflow representation in the CAKE system [9] uses an object-oriented representation (originally developed for cases) for ontologies and metadata annotation. Tasks can be organized in a hierarchy of classes, each of which contains certain properties of a task, which can be inherited from the super class. For example, the *Assign room* task includes a role description stating that the assignment must be performed by the responsible administrative department. Another property may state the average duration for task execution.

The semantic workflow representations in Wings augment traditional workflow representations with semantic constraints that indicate metadata properties of workflow tasks and datasets [20, 21]. Each workflow constituent has a workflow variable associated with it, and the semantic constraints express constraints over those variables. In the case of data variables, the constraints are

expressed in terms of metadata properties. For example, an input dataset may be restricted to be a discrete dataset, and if so then its variable would have a constraint stating that it has the property of being discrete. In the case of components, the constraints express how the execution of the software component changes the metadata properties of the input datasets into output datasets. Wings uses a workflow reasoning algorithm to enrich workflows by propagating the constraints throughout the workflow structure [20]. This algorithm takes simple workflow graphs created by users and automatically augments them with semantic information by adding and propagating semantic constraints that can be used for subsequent retrieval, as we propose in this paper.

For semantic workflow representation some languages that include semantic constraints expressed in OWL have been proposed, such as OWL-S, WSMO, SWSL, and SWSF [2, 42]. These languages include task decomposition structures and complex precondition expressions. Our CAKE and Wings approaches could be used with any of those languages, though only a small subset of the constructs allowed in those languages are used in our systems.

2.3. Workflow similarity

The notion of similarity plays a key role in CBR, since cases are selected based on their similarity to the current problem. While early CBR approaches were usually restricted to syntactic similarity measures (such as inverse Euclidean or Hamming distances), the current view is that the similarity measure should consider as much semantic as possible. Consequently, similarity measures must be modeled as part of the knowledge acquisition process during CBR application development.

Similarity is usually formalized in terms of a function that maps a pair of problem descriptions to a real number, often restricted to the unit interval $[0, 1]$, with the convention that a high value indicates a high similarity. Further, similarity is linked with the notions of preference and utility [41, 6, 10]: A higher degree of similarity suggests that a case is more useful for solving the current problem and hence should be preferred.

As a means for practical modeling of similarity functions, the so-called *local-global principle*, first proposed by Richter (for details see [41, 31, 6]) is widely used. Modeling similarity means decomposing the similarity function according to certain properties of the case. A *local similarity function* is defined for each individual property, reflecting the utility of a case with respect to the single property only. The local similarities are then aggregated into the *global similarity* by means of an *aggregation function*. This function appropriately combines all local similarity values (e.g. by a weighted average) into a single value that aims at reflecting the utility of the case as a whole. Besides our own system CAKE, many academic and commercial CBR tools (e.g. myCBR[2] from DFKI,

[2] http://www.mycbr-project.net/

jColibri from Universidad Complutense de Madrid [3], IAS[4] from EMPOLIS Information Management GmbH) use this approach for similarity assessment. It has proven successful in many academic and industrial CBR applications with relational or object-oriented case representations [7].

Workflows, however, require a graph-based representation to appropriately cover control- and dataflow [27, 18, 15, 30, 34, 38]. This introduces structural aspects into the representation and thereby makes the similarity assessment much more complicated. Several graph algorithms have been proposed for similarity assessment such as sub-graph isomorphism, maximal common sub-graphs, or edit-distance measures [26, 25, 40, 22, 38, 30, 18], which usually cause problems due to their high computational complexity. To avoid these problems, previous work investigated structure-less workflow retrieval based on query and workflow representation being plain textual descriptions, sets of (social) tags [22], or abstract workflow features [9]. Other work considers the graph structure (or parts of it), but uses standard syntactical similarity measures [38, 30, 23, 18, 16]. Previous work on Wings uses semantically annotated graph structures, but a crisp matching approach rather than similarity [20] for retrieval. Recent work by Montani et al. [40, 39] describes an edit trace distance measure for sequential workflow execution traces, which makes use of taxonomic distances and thus can be considered a step towards a semantic similarity measure for workflows. Also the SAI toolkit [27] enables workflow retrieval using domain-specific similarity measures, thus providing in principle also the opportunity to use semantic similarity measures.

2.4. Workflow retrieval

For workflow retrieval, several CBR and related approaches exist today. The CODAW system [34] supports incremental modeling of workflows by a similarity-based reuse of the workflow templates using an HTN planner that employs an inexact, graph-based matching. Leake et al. [30] evaluate the execution paths of past workflows in order to support user extension of workflows that are under construction. Their generic SAI toolkit [27] performs retrieval in two steps to control retrieval cost. Retrieval and clustering of workflows for business process monitoring was proposed by Montani et al. [39]. Recently, a probabilistic similarity model for workflow execution paths was proposed by Becker et al. [4]. The myGrid project[5] focusses on the discovery, reuse and repurposing of bioinformatics workflows [22]. They distinguish between direct *re-use* of an existing workflow as is and *re-purposing* an existing workflow by modifying some aspect of it. Goderis [22] defines requirements and bottlenecks for workflow re-use based on many user interviews and practical experiences.

[3] http://gaia.fdi.ucm.es/research/colibri
[4] http://www.empolis.com
[5] www.mygrid.org.uk

3. Graph-based framework for workflow similarity

We now describe a new approach for representing semantic workflows using graphs that is particularly suitable to enable the modeling and computation of semantic similarity measures. The described representation scheme is quite general as it allows to represent different kinds of workflows, including business workflows and scientific workflows.

3.1. Representation of semantic workflows

In line with previous work on graph-based workflow representation [14, 18], we represent workflows as semantically labeled directed graphs. A *semantic workflow graph* W is a quadruple $W = (N, E, S, T)$ where N is a set of nodes and $E \subseteq N \times N$ is a set of edges. $T : N \cup E \to \Omega$ associates to each node and each edge a *type* from Ω (see below). $S : N \cup E \to \Sigma$ associates to each node and each edge a *semantic description* from a semantic metadata language Σ. We do not demand a particular language for Σ, but just assume some language for semantic metadata for which similarity measures can be defined (see Sect. 3.3). However, the definition of semantic workflow graphs includes the following definition of Ω, specifying the available types of nodes and edges as well as restrictions on T concerning the usage of those types:

A semantic workflow graph (and hence Ω) contains the following types of nodes:

- Each workflow consists of exactly one **workflow node**. The semantic description associated with a workflow node represents important general properties of the entire workflow. This can be a workflow classification in some ontology, a set of semantic tags, or other properties related to the quality, performance, or reliability of the workflow.

- Each task in a workflow is represented by a **task node**. Its semantic description typically classifies the task in some task ontology and may provide additional functional properties of the task. It also includes the description of workflow roles (i.e., human agents or services) for the execution of those tasks.

- Each data item in a workflow is represented by a **data node**. Its semantic description classifies the data item within some data type ontology and may specify certain data properties (e.g., if a data set has missing values) relevant for retrieval. Control-flow centric workflows (Fig. 1a) may have no data nodes.

- Each control flow element in a workflow, such as split/join elements for and/xor blocks, are represented as a **control-flow node**. The semantic description of a control flow node specifies which control flow element is represented. Data-centric workflows (e.g. Fig. 1b) may have no control flow nodes.

In addition, a semantic workflow graph (and hence Ω) contains the following types of edges:

- The workflow node is linked to each of the other nodes by a **part-of edge**. The semantic description of such an edge specifies the role of the associated node within the overall workflow. Data nodes can be linked with the workflow node via edges having one of the following semantic description: *is-input*, *is-output*, *is-intermediate*, *is-parameter*. For example, in Fig. 1b *inputData1* is overall workflow input data and hence linked with *is-input*. *inputData* and *outputData* in Fig. 1b are data produced and further processed within the workflow and are hence linked with *is-intermediate*. Further, the semantic descriptions for part-of edges include *has-task* for task nodes linked to a workflow node and *has-control* for linking to control-flow nodes.

- The dataflow among tasks is represented using **dataflow edges**. They link data nodes to task nodes or vice versa. The semantic description of such an edge indicates whether the data item was *consumed* as input data or *produced* as output data by the task. All arrows shown in Fig. 1b are turned into dataflow edges in the graph.

- The control-flow among tasks is represented using **control-flow edges**. Such an edge connects two task nodes or a task node with a control-flow node. An edge from node $n1$ to $n2$ indicates that node $n2$ must be executed after node $n1$. All arrows shown in Fig. 1a will be turned into control-flow edges in the graph.

- Semantic constraints among properties of data nodes are represented using **constraint edges** that connect two data nodes. The semantic description includes the definition of a constraint. For example, a constraint may state that test and training data must come from the same domain.

The so defined semantic workflow graph can be used to represent different kinds of workflows, such as scientific or business workflows. Figure 2 shows the graph representation of the workflow from Fig. 1b. It contains one workflow node ($n1$), nine data nodes ($n2 - n10$), and three task nodes ($n11$ - $n13$) as well as the respective edges. A simplified fraction of some semantic descriptions are shown in grey boxes.

3.2. Workflow repository and queries

A repository of successful workflows reflecting best-practice in a domain is the core of a POCBR approach. In case-based reasoning terminology, this repository is the case base that stores the available experience. Given the semantic workflow graph representation, we represent a workflow repository as a set of semantic workflows over the same semantic metadata language Σ. Hence, a workflow repository is always tied to a particular domain and its semantic

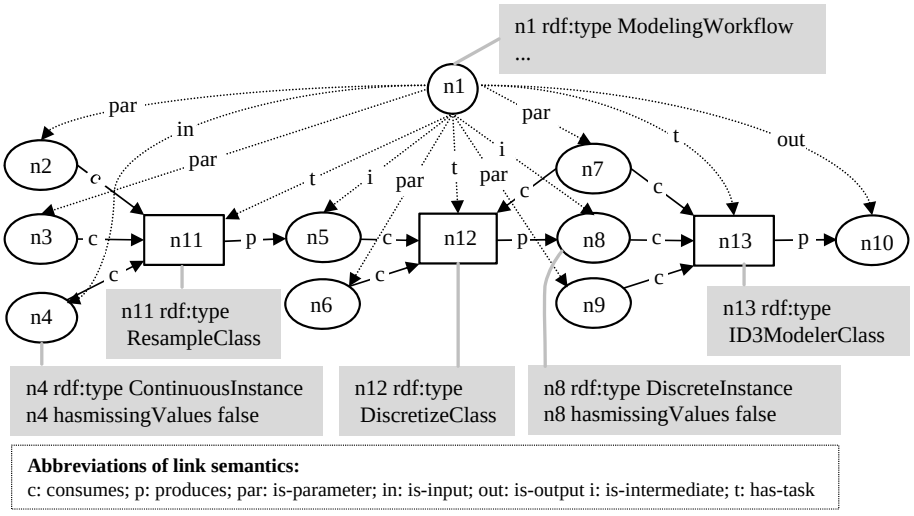

Figure 2: Fraction of a workflow graph.

metadata representation. Thus, we define a case base $CB = \{CW_1, ...CW_n\}$ as a set of cases CW_i each of which is a semantic workflow graph over Σ.

Users can query this case base with a specification of important properties of the workflow s/he wants to create in order to retrieve potentially reusable workflows. Previous work on workflow reuse and discovery has already addressed the question how typical queries look like. For scientific workflow, Goderis et al. [23] studied the importance of different criteria for workflow discovery. They identified that the task and data items that occur in the workflow are relevant as well as general characteristics of the workflow related to quality, performance, and reliability. The workflow structure is also very important, particularly the dataflow and control flow as well as the used control-flow constructs. Gil et al. [20] confirm these results and develop different query types for scientific workflows that enable to combine structural workflow properties with the ontological information concerning data and task items. Also for business workflows, the relevant types of queries have been identified and different query languages have been proposed by Beeri et al. [5] and Awad [3]. This work clearly shows that it is useful to construct queries in the same way as workflows are constructed. Queries can be patterns built from connected workflow elements, which are then matched against the workflows in the repository.

In the light of these results, we focus on queries that are represented as a semantic workflow graph. Such a query specifies some workflow elements together with the semantic description that are considered requirements on the workflows to be retrieved. A simple query could even consist solely of a workflow node that contains a semantic descriptions specifying the class of workflow and some general properties, such as quality requirements. More sophisticated queries may in addition contain some unconnected data and/or task nodes that spec-

ify that these nodes should occur in the workflow one is looking for. Structural properties related to the data and/or control flow can be specified by linking the nodes by contol-flow and/or dataflow links, thus forming a partial workflow (or even several unconnected partial workflows). Such queries specify the need for workflows that contain the mentioned sub-workflows or that contain something similar to the mentioned sub-workflows in the query.

For example, a query for a scientific workflow could state that someone is looking for workflows that first discretize some continuous data and then use the discrete data in a decision tree modeler. Hence, the query (see Fig. 3) would contain a workflow node ($n21$) as well as two data nodes: one specifying the continuous data ($n22$) and one specifying the discretized intermediate data ($n24$). Further, it includes two task nodes, one that specifies a task that can discretize data ($n23$) and one that specifies a decision tree modeler task ($n25$).

The similarity-based retrieval we aim at, could then retrieve for example the workflow shown in Fig. 1b, which is a suitable match as it contains an ID3 modeler, which is a specific decision tree modeler. In case the query would have asked for a Bayes modeler, the workflow from Fig. 2 could be potentially useful as well, but then the user has to modify the used modeler task, which should be reflected in a lower similarity for this query.

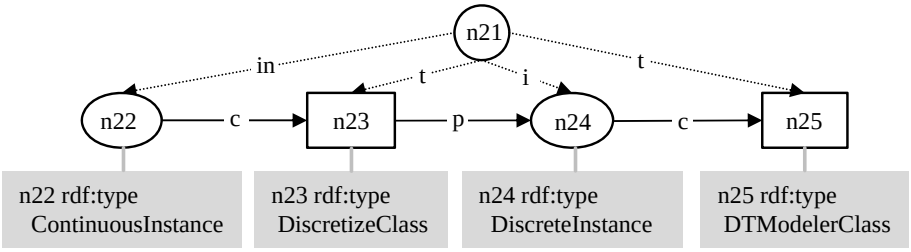

Figure 3: Fraction of a workflow graph.

3.3. Modeling workflow similarity

In order to assess the similarity of a case workflow $CW = (N_c, E_c, S_c, T_c)$ wrt. a query $QW = (N_q, E_q, S_q, T_q)$, we need to consider the constituents of the workflow as well as the link structure. This requires similarity models that enable to compare the workflow elements. We now present a new similarity model, which can be considered an enhancement of the well-known local/global approach for structural CBR [13, 6]. The local similarity measures assess the similarity between two nodes or two edges of the same type. The global similarity for workflows is obtained by an aggregation function combining the local similarity values within a graph mapping process.

3.3.1. Similarity of semantic descriptions

The local similarity assessments are based on the semantic descriptions of the nodes and edges. Therefore, we define a similarity function:

$$sim_\Sigma : \Sigma \times \Sigma \to [0, 1].$$

The detailed formalization of this similarity measure depends on the language Σ and can itself be modeled using the local/global approach aggregating local similarity measures for the individual properties in the semantic description. In our work, we treat the semantic descriptions in an object-oriented fashion and use well established similarity measures [11, 6]. This enables us to model appropriate local similarity measures for simple properties such as *hasmissingValues* shown in Fig. 2. Further, we use class hierarchies for tasks and data objects (see Fig. 4 for an example) as a basis to model similarity measures.

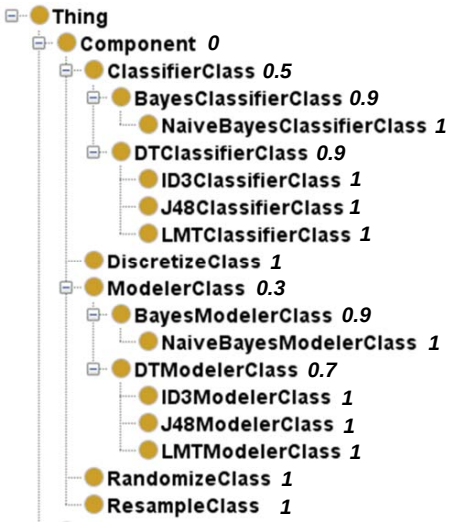

Figure 4: Fraction of the task ontology with similarity values s_{cl} assigned to classes cl.

For example, the similarity of the semantic descriptions of the nodes $n25$ and $n13$ is computed by the following similarity measure that uses the shown task ontology:

$$sim(cl_q, cl_c) = \begin{cases} 1 & \text{if } cl_q \sqsubseteq cl_c \text{ or } cl_q \sqsupseteq cl_c \\ s_{<cl_q, cl_c>} & \text{otherwise} \end{cases}$$

This similarity measure compares the class of a task in a query cl_q with a class of a task in the case cl_c. If one is subclass of the other, the similarity is 1. Otherwise, the similarity value $s_{<cl_q, cl_c>}$ results, where $< cl_q, cl_c >$ denotes the least common superclass of cl_q and cl_c. The similarity values s_{cl} assigned to the classes of the ontology (see Fig. 4) are determined as part of the similarity modeling. According to this measure, the similarity between the semantic descriptions of $n25$ and $n13$ would be 1; the similarity between *BayesClassifierClass* and *ID3ClassifierClass* would be 0.5 as this is the value assigned to the common superclass *ClassifierClass*.

3.3.2. Similarity of nodes and edges

Based on the similarity of the semantic descriptions, we define the similarity of nodes and edges. Nodes with different types are considered dissimilar; there similarity is always zero. The similarity of nodes of equal type is defined as the similarity of the semantic descriptions. Hence, we define node similarity $sim_N(n_q, n_c)$ for $n_q \in N_q$ and $n_c \in N_c$ as follows:

$$sim_N(n_q, n_c) = \begin{cases} sim_\Sigma(S_q(n_q), S_c(n_c)) & \text{if} \quad T_q(n_q) = T_c(n_c) \\ 0 & \text{otherwise} \end{cases}$$

Edge similarity is more sophisticated as it should consider not just the semantic description of the edges being compared, but also the nodes that are linked by the edges. For example, two dataflow edges should only be considered similar if they link similar data objects to similar tasks. To reflect such considerations, we define edge similarity $sim_E(e_q, e_c)$ for $e_q \in E_q$ and $e_c \in E_c$ as follows:

$$sim_E(e_q, e_c) = \begin{cases} F_E \begin{pmatrix} sim_\Sigma(S_q(e_q), S_c(e_c)), \\ sim_N(e_q.l, e_c.l), \\ sim_N(e_q.r, e_c.r) \end{pmatrix} & \text{if} \quad T_q(e_q) = T_c(e_c) \\ 0 & \text{otherwise} \end{cases}$$

For an edge e, the expression $e.l$ denotes the left node (origin) of the edge and $e.r$ denotes the right node (destination). The function $F_E(S_e, S_l, S_r)$ is an *aggregation function* that combines the semantic similarity S_e between the edges and the two similarities S_l and S_r of the left and right nodes to the overall similarity value. In our implementations, we define F_E as follows: $F_E(S_e, S_l, S_r) = S_e \cdot 0.5 \cdot (S_l + S_r)$.

3.3.3. Similarity of workflows

The overall workflow similarity between QW and CW is now defined by means of a *legal mapping*, i.e., a type-preserving, partial, injective mapping $m : N_q \cup E_q \to N_c \cup E_c$ that satisfies the following five constraints:

$$\begin{array}{lll} T_q(n_q) = T_c(m(n_q)) & T_q(e_q) = T_c(m(e_q)) & \\ m(e_q.l) = m(e_q).l & m(e_q.r) = m(e_q).r & \forall x, y\ m(x) = m(y) \to x = y \end{array}$$

Please note that in such a legal mapping m, a case node or edge can only be the target of one query node or edge, respectively. Since the mapping can be partial, not all nodes and edges of the query must be mapped to the respective case elements. Also an edge can only be mapped if the nodes that the edge connects are also mapped to the respective nodes which are linked by the mapped edge. For example, a legal mapping m_1 for the shown workflow and query is:

$$\begin{array}{lll} n21 \mapsto n1 & n22 \mapsto n5 & n23 \mapsto n12 \\ n24 \mapsto n8 & n25 \mapsto n13 & (n21, n22) \mapsto (n1, n5) \\ (n21, n23) \mapsto (n1, n12) & (n21, n24) \mapsto (n1, n8) & (n21, n25) \mapsto (n1, n13) \\ (n22, n23) \mapsto (n5, n12) & (n23, n24) \mapsto (n12, n8) & (n24, n25) \mapsto (n8, n13). \end{array}$$

For a particular mapping m, the similarity $sim_m(QW, CW)$ between QW and CW is defined by aggregating the local similarity values of the nodes and edges mapped by m. Let $\{n_1, \ldots, n_u\} = N_q \cap \mathrm{Dom}(m)$ be the set of query nodes mapped by m and let $\{e_1, \ldots, e_v\} = E_q \cap \mathrm{Dom}(m)$ be the set of query edges mapped by m. Here, $\mathrm{Dom}(m)$ denotes the domain of m. We then determine the vector of local node similarities $(sn_1, \ldots, sn_u)$ by computing $sn_i = sim_N(n_i, m(n_i))$ $(i = 1 \ldots u)$ as well as the vector of local edge similarities $(se_1, \ldots, se_v)$ by computing $se_i = sim_E(e_j, m(e_j))$ $(j = 1 \ldots v)$. The following aggregation function F_W combines them into an overall similarity value with respect to the mapping m:

$$sim_m(QW, CW) = F_W((sn_1, \ldots, sn_u), (se_1, \ldots, se_v), |N_q|, |E_q|)$$

The parameters $|N_q|$ and $|E_q|$ enables F_W to consider partial mappings appropriately, i.e. nodes and edges not mapped should not contribute to the overall similarity. In our implementation we define F_W as follows:

$$F_W((sn_1, \ldots, sn_u), (se_1, \ldots, se_v), n_N, n_E) = \frac{sn_1 + \cdots + sn_u + se_1 + \cdots + se_v}{n_N + n_E}$$

Please note that each mapping m can be interpreted as a particular suggestion for the reuse of the case workflow: the mapped case nodes and edges should be considered solution elements for the respective nodes and edges in the query. The similarity value for the mapping assesses the utility of this reuse opportunity. For example, the similarity of the above mentioned mapping m_1 would yield a similarity of 1, since all query nodes are mapped to perfectly matching case nodes. Mapping the nodes in a different manner, however, leads to node and edge mappings with lower similarity (e.g. if $n23$ would be mapped to $n11$), which indicates a less useful way to reuse the workflow.

For a single case there usually exist several legal mappings. Each mapping reflects a particular way of reusing the case. The overall workflow similarity $sim(QW, CW)$ should correspond to the best possible mapping (see also [14]), i.e. the mapping with the highest similarity to the query:

$$sim(QW, CW) = \max\{sim_m(QW, CW) | \text{ mapping } m\}.$$

Given that in the example m_1 is a mapping with similarity of 1, the overall similarity is also 1. Please note that this is only the case, because the semantic similarity measure for tasks takes into account that the ID3 Modeler is a specific decision tree modeler, which is hence considered a perfect match. This is just one example, of how the presented similarity model can be configured to reflect reusability in a particular domain. In general, the similarity model is defined by three *model parameters*:

1. the similarity function sim_Σ for semantic descriptions,
2. the aggregation function for edges F_E, and
3. the aggregation function for workflows F_W.

3.4. Experimental evaluation

We now focus on the question whether the computed similarity values are appropriate within the context of a POCBR application aiming at retrieving workflows for reuse. For this purpose, the similarity measures should be able to approximate the utility of the cases wrt. the problem formulated within the query (see [6], p.94*ff.*) in the context of the concrete application. Moreover, a good and broadly applicable similarity model should enable to flexibly determine appropriate model parameters for many applications. Whether this is the case, can of course only be assessed by analyzing a series of real-world applications. As a first step towards this goal, we developed two initial workflow retrieval applications in two complementary domains: *administrative business workflows* (similar to Fig. 1a) and *scientific data mining workflows* (similar to Fig. 1b). Based on our previous work [37, 20], we developed for each domain an ontology for workflows, tasks (and data), as well as a case base of 20 workflows.

The aim of the first experiment is to show, whether it is possible to model semantic similarity measures that approximate the utility assessment of a human user. More precisely, the hypothesis is:

H1: *The similarity model enables to define model parameters such that the computed semantic similarity is better aligned with an expert's assessment than a standard lexical similarity measure.*

For each domain, we determined 10 queries that are related to the 20 cases in the case base. For each query, a human domain expert was asked to select the 5-6 best suitable cases in the case base and to determine a partial order among those best cases. For each domain, a semantic similarity model was developed by manual knowledge acquisition and modeling. This includes specific similarity measures sim_Σ considering data and task ontologies as well as several similarity measures to compare node and task properties. For each query, we computed the similarity of each case in the case base by applying an exhaustive search algorithm that computes all mappings, thereby producing an optimal solution. As baseline for the comparison, we also computed the similarity sim_Σ by a lexical similarity measure, in particular a Levenshtein similarity measure on the task and data names. Both similarity measures use the same aggregation functions F_E and F_W.

For both similarity measures, we compared the ordering of the cases produced with respect to the two similarity measures to those of the human expert. As a measure of correctness, we used the ranking measure proposed by Cheng et al. [15]: they define the *correctness* and *completness* of an ordering $\succ$ with respect to a reference order $\succ_*$ based on the concordance and discordance of the two orders. The concordance C of the two orders is the number of case pairs ordered equivalently by both orders and the discordance D is the number of case pairs for which both orders differ. Based on this, correctness and completeness are defined as follows:

$$Corr = \frac{C - D}{C + D} \quad \text{and} \quad Compl = \frac{C + D}{|\succ_*|}.$$

Similarity	Administrative business workflows			Scientific data mining workflows		
	Correct	Complete	Retrieval Time	Correct	Complete	Retrieval Time
Semantic	0.708	0.910	24.00 sec	0.840	0.693	8.00 sec
Lexical	0.542	0.911	24.36 sec	0.593	0.751	7.78 sec

Figure 5: Similarity Measure Evaluation.

In our evaluation, we determine for each of the 10 query workflows QW the order on the 20 cases through: $CW_1 \succ CW_2$ iff $sim(QW, CW_1) > sim(QW, CW_2)$. The partial reference order $\succ_*$ is given by the human ranking for the same query. The value for correctness is within the interval [-1,1]. If it has a value of 1 then every ordering stated in $\succ$ is correct, if it has a value of -1, every ordering stated in $\succ$ contradicts the reference order. A value of 0 indicates that there is no correlation between both orders. The completeness value is within the interval [0,1]. It punishes the abstention from comparisons, i.e. situations in which the similarity of two cases is equal although the human expert prefers one case over the other. The higher the completness value, the more orderings (correct or wrong) are contained in $\succ$. Figure 5 shows the results of the evaluation for the semantic similarity measure compared to the Levenshtein measure for both domains. The averaged correctness and completeness values over all 10 queries are displayed. The figure clearly shows an improved correctness for the semantic similarity measure in both domains, while the completeness slightly suffers for the data mining workflows. This clearly confirms the hypothesis H1.

Additionally, Fig. 5 shows the average retrieval time over all 10 queries using linear retrieval over the full case base, applying exhaustive search for similarity assessment. The similarity computation was implemented in SWI-Prolog[6] and the computation time was determined on a Dell Optiplex 980 desktop computer with an Intel quad-core i5-750 CPU @ 2.66 GHz, using 8 GB Ram and running the 64-bit version of Windows 7. These figures clearly show that this approach is computationally unacceptable for practical applications. This motivates our work on a more scalable framework, described in the next section.

4. Similarity computation and workflow retrieval

While similarity computation by exhaustive search guarantees to find the optimal match, it is computationally not feasible. Greedy search is proposed as alternative search algorithm for similarity assessment of workflow graphs [30, 18, 14] as it enables to quickly find a local optimum. However, the resulting error can hardly be controlled as the local optimum can differ significantly from the global optimum, hence producing too low similarity values. The A* search algorithm promises to be good alternative over the above mentioned approaches,

[6]The multi-threaded 64-bit Version 5.10.2 was used.

given a well-informed admissible heuristic function can be determined. In the following, we will develop several A* search variants and demonstrate their performance and similarity error in experimental evaluations.

4.1. Similarity computation by A search*

The A* algorithm (see, for example, Russel et al. [43] for an introduction) maintains a priority queue of open search nodes. In each step, the first (best) node in the queue is removed. If it represents a solution, A* terminates. Otherwise this node is expanded, i.e. each successor node in the search is determined and inserted into the priority queue. When applied for finding the best match m between a query and a case graph elements, a search node S basically represents the current mapping $S.m$ as well as the not yet mapped nodes $S.N$ and edges $S.E$. During node expansion, the mapping $S.m$ is extended in all possible ways by one additional mapping of a node or edge. The order in which the expanded nodes are inserted into the priority queue is essential for A*. Therefore, each search node S is evaluated by a function $f(S) = g(S) + h(S)$. In the traditional formulation, A* aims at minimizing cost, hence $g(S)$ are the cost already occurred and $h(S)$ is a heuristic estimation function for the remaining cost to the solution. As we apply A* for maximizing the similarity value, the functions must be interpreted differently, i.e. $g(S)$ is the similarity of the current mapping S, while $h(S)$ is an heuristic estimation of the additional similarity that can be achieved through the mapping of the remaining nodes and edges. Nodes are inserted into the priority queue in decreasing order of $f(S)$, i.e., the search node with the highest f-value is expanded first. To achieve an admissible heuristic estimation function, $h(S)$ must be an upper bound of the similarity. In the A* algorithm shown below, the value $f(S)$ is also stored in the search node, noted as $S.f$. The A* search algorithm (divided into the toplevel looping algorithm and the separate expansion function) is called with a query workflow QW and a case workflow CW and returns the similarity value.

```
A*Search(QW = (N_q, E_q, T_q, S_q), CW = (N_c, E_c, T_c, S_c))
{  S_0.N = N_q;   S_0.E = E_q;   S_0.m = ∅;   S_0.f = 1;   Q =< S_0 >;
   while first(Q).N ≠ ∅ or first(Q).E ≠ ∅
         do  { Q = Expand(Q) };
   return(first(Q).f);
}

Expand(Q)
{  S = first(Q); Q = rest(Q); x_q =select(S);
   forall x_c ∈ E_c ∪ N_c s.th. the mapping S.m ∪ (x_q, x_c) is legal
   do { S'.m = S.m ∪ (x_q, x_c); S'.N = S.N \ {x_q}; S'.E = S.E \ {x_q};
        S'.f = sim_{S'.m}(QW, CW) + h(S');
        Q =insert(S', Q);
      }
   Return Q;
}
```

Here, *first*(Q) is the first priority queue node, *rest*(Q) removes the first node and returns the rest, and *insert*(S', Q) inserts the new node S' into Q according to the f-value. During insertion, the maximum size of the queue can be restricted (maximum queue size) to cut some branches of the search tree to improve performance on the risk of loosing global optimality. The *select* function determines the next node or edge to be expanded. x_q can be either a query node or edge.

We developed two versions of A* with different estimation and select functions. The first version $A*I$ uses a naive estimation function which assesses the similarity of each not yet mapped node or edge as 1. Assuming the aggregation function F_W shown in Sect. 3.3, the contribution of each not mapped element to the overall similarity is computed by h_I. The select function first matches all nodes, before the edges are selected. Which element from $S.N$ or $S.E$ is selected is not determined; we choose randomly according to an internal node/edge id.

$$h_I(S) = \frac{|S.N|+|S.E|}{|N_q|+|E_q|}$$

$$select_I(S) = \begin{cases} n_q \in S.N & \text{if } S.N \neq \emptyset \\ e_q \in S.E & otherwise \end{cases}$$

The second version $A*II$ uses a better informed admissible heuristic. For each not mapped query node or edge it determines the maximum possible similarity a mapping can achieve independent of the mapping of the other nodes or edges. These values can be computed prior to search and cached. Also, it aims at matching edges as soon as possible. As matching edges requires the connecting nodes being matched already, the edge expansion leads to a low branching factor. It is 1 if between two nodes there is at most one edge per type, which is the case in our two example domains. Hence, the size of the queue does not increase, while the accuracy of $f(S')$ increases.

$$h_{II}(S) = \sum_{x \in S.N \cup S.E} \left(\max_{y \in N_c \cup E_c} \{sim_{E/N}(x, y)\} \right) \cdot \frac{1}{|N_q|+|E_q|}$$

$$select_{II}(S) = \begin{cases} e_q \in S.E & \text{if } e_q.l \notin S.N \text{ and } e_q.r \notin S.N \\ n_q \in S.N & otherwise \end{cases}$$

It is well known that the A* algorithm in general finds the optimal value, if an admissible heuristic is used. Hence, the two proposed algorithms for similarity computation will compute the exact similarity $sim(QW, CW)$ according to the definition given in Sect. 3.3.3. However, to deal with the large memory consumption of A*, it is often required to introduce a limit for the queue size. In this case the search space is pruned and it cannot be guaranteed that the optimal solution is found. Hence, a similarity error may occur in the sense that the computed similarity is lower than the exact value. During the empirical evaluation of the algorithms, the similarity error must be considered in addition to the computation time for retrieval (see Sect. 4.3).

4.2. Parallelized A retrieval*

While the described A* algorithms aim at improving performance of a single similarity assessment, linear retrieval with large case bases will still require one run of the search algorithm per case in the case base. To ensure better scalability with growing case bases, we now propose a parallelized A* II variant, called *A*P*. It enables to compute the top k cases from the case base without fully computing the similarity for all cases. To this end, the search process is parallelized for all cases of the case base, maintaining one individual queue Q_i for each case. In every step, the node from the queue with the highest f-value from all queues of still ongoing search processes is expanded. Search terminates, when at least k searches have terminated and when the similarity of the k-best case is higher than all f-values of the remaining queues. Since the f-values are upper bounds for the final similarity, it is ensured that none of the remaining cases can ever exceed the similarity of the known k-best case. Hence, completeness of k-best retrieval is guaranteed. The following algorithm returns the list of the k-best cases (and possibly some more) together with its similarity value.

*A*P Retrieval*$(QW = (N_q, E_q, T_q, S_q), CB = \{CW_1, \ldots, CW_n\}, k)$
{ $S_0.N = N_q$; $S_0.E = N_E$; $S_0.m = \emptyset$; $S_0.f = 1$;
for $i = 1 \ldots n$ **do** { $Q_i =< S_0 >$ } ;
res $= \emptyset$;
repeat
{ $j = \arg\max_{i \in \{1 \ldots n\} \setminus \text{res}} \{\mathit{first}(Q_i).f\}$;
$Q_j =$*Expand*(Q_j) ;
if *first*$(Q_j).N = \emptyset$ **and** *first*$(Q_j).E = \emptyset$ **then** res $=$ res $\cup \{j\}$;
} **until** $|res| \geq k$ **and** k-th(*first*$(Q_i).f | i \in$res$) \geq \max_{j \in \{1 \ldots n\} \setminus \text{res}} \{\mathit{first}(Q_j).f\}$;
return $\{(\mathit{first}(Q_i).f, i) \mid i \in$res$\}$
}

4.3. Experimental evaluation

For experimental purposes, we implemented the three A* variants in SWI-Prolog and evaluated them with respect to computation time. As a baseline for this comparison, we also included the exhaustive search used in the first experiment as well as a standard greedy search algorithm. For the experimental evaluation, we formulated the following four hypotheses:

H2a: *The average retrieval time of A*I is shorter than that of exhaustive search.*

H2b: *The average retrieval time of A*II and A*P are shorter than that of A*I.*

H2c: *The average similarity error of A*I,II,P are lower than that of greedy search.*

H2d: *The average retrieval time of A*P is lower than that of A*II, if* $k << |CB|$.

We tested the hypotheses for the two workflow domains. The similarity models and the case base of 20 cases from Sect. 3.4 are used. To assess the retrieval performance, we used 30 queries for each domain: the 10 queries from Sect. 3.4 plus the 20 cases of the case base itself. We determined the average retrieval time per query as well as the average similarity error of the retrieved cases. As a base line for this, we determined for each query and case the optimal similarity value by running the exhaustive search algorithm with a time limit of two hours per similarity computation.

Retrieval Method	Qsize	k	Retrieval time [sec]	Similarity Error
Exhaustive			795.269	
Greedy	1	20	0.112	0.221
A* I	10	20	0.279	0.092
	100	20	3.829	0.102
	300	20	21.075	0.091
A* II	1	20	0.254	0.023
	10	20	0.792	0.011
	100	20	15.051	0.005
A* P	1	5	0.226	0.018
	10	5	0.584	0.008
	100	5	9.332	0.004

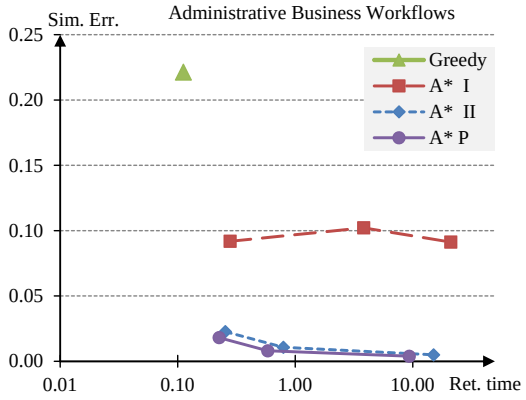

Figure 6: Retrieval performance: Administrative business workflows

Retrieval Method	Qsize	k	Retrieval time [sec]	Similarity Error
Exhaustive			937.500	
Greedy	1	20	0.028	0.100
A* I	10	20	0.415	0.090
	100	20	5.244	0.069
	300	20	19.937	0.055
A* II	1	20	0.345	0.037
	10	20	0.852	0.020
	100	20	7.093	0.007
A* P	1	5	0.311	0.005
	10	5	0.667	0.007
	100	5	5.663	0.004

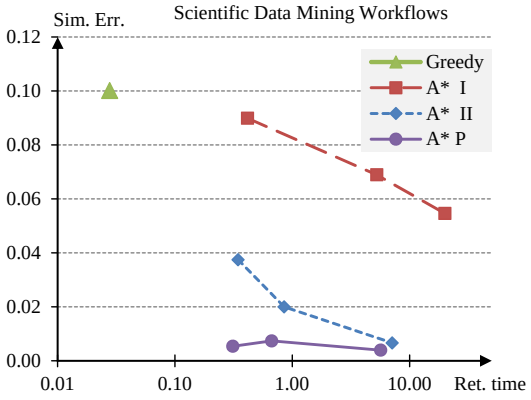

Figure 7: Retrieval performance: Scientific data mining workflows

Figures 6 and 7 show the results for the two domains for the different A* variants with different limits on the queue size. A*P is evaluated for $k = 5$ (out of 20 cases). For the other algorithms the performance does not depend on the number of cases to be retrieved. We used the same Dell Optiplex computer as in the first experiment.

The graphs plot the similarity error over the retrieval time (logarithmic scale). Increasing the queue size limit clearly leads to an increased retrieval time, but also to a reduced error. The figures clearly indicate that the hypotheses H2a,b and c are confirmed. Concerning H2d, the advantage of A*P over A*II is not very dominant. Therefore, we performed an additional experiment with a case base of 203 cases from the scientific data mining domain used in [20]. Figure 8 shows the average retrieval time for different values of k. It clearly confirms hypothesis H2d.

	Retrieval Time [sec]		
Qsize	A* P, k=5	A* P, k=10	A* II
10	0,386	0,483	3,04
100	0,667	3,98	14,15

Figure 8: Retrieval performance: Scientific data mining workflows, 203 cases

4.4. Integration into CAKE

The presented evaluation so far has been conducted using an experimental implementation of the proposed retrieval algorithms in SWI-Prolog. The purpose of this implementation was primarily to ease the algorithm development and to empirically investigate their properties. In order to approach its use in realistic POCBR applications, we re-implemented the A*P retrieval (which clearly outperforms the other variants) in JAVA and integrated it into CAKE [9, 38] .

CAKE[7] is a software platform being developed at the University of Trier during the past 7 years. It has been used to build various prototypical applications involving an integration of CBR and agile workflows technology, in various areas such as fire services [19], chip design [38], medicine [35], geographical information systems [19], administrative business workflows, and cookery [36]. CAKE is a generic system that is fully implemented in JAVA and which can be configured by XML files that encode ontologies, similarity measures, case bases, and workflows. It includes a CBR component that represents cases using configurable object-oriented domain models. It has a large library of configurable similarity measures that enable to define semantic similarity measures for object-oriented representations. Further, CAKE includes an agile workflow engine that supports the execution of agile workflows.

To evaluate the performance of CAKE's new retrieval component, we repeated the tests with the A*P retrieval with the same domains, similarity measures, and case bases on the same DELL Optiplex computer. Figure 9 shows a comparison between the retrieval time of the initial SWI-Prolog implementation and the new CAKE implementation in JAVA. The figure states the average retrieval time per query in seconds for different domains and parameters.

[7] http://cake.wi2.uni-trier.de

Domain	Qsize = 10		Qsize = 100	
	PROLOG	CAKE	PROLOG	CAKE
Administrative business workflows, 20 cases, k=5	0.584	0.040	9.332	0.199
Scientific data mining workflows, 20 cases, k=5	0.667	0.038	5.663	0.100
Scientific data mining workflows, 203 cases, k=5	0.386	0.023	0.667	0.027
Scientific data mining workflows, 203 cases, k=10	0.483	0.025	3.98	0.034

Figure 9: Comparison of average retrieval time per query in seconds.

The results demonstrate clearly that we succeed in improving the initial implementation by a factor of approximately 15-115 (depending on the parameters). Given that the current retrieval time is less than 200 ms in every case, we are quite optimistic that the developed retrieval approach is computationally feasible also for realistic applications. In future work, we will try to demonstrate this in new domains using large workflow repositories.

5. Conclusion and Future Work

This paper contributes to the core problems of process-oriented case-based reasoning, particularly to the representation of semantically annotated workflows, the similarity assessment of workflows considering the semantic annotation, and the efficient similarity-based retrieval of workflows from a workflow repository. As the presented approach does not rely on a particular language for semantic annotations and does not impose restrictions on how local similarity measures and aggregation functions can be defined, it is quite generally applicable to various domains. We demonstrated this by its application to two different workflow domains with very different characteristics. Besides this, the proposed modeling approach for similarity measures allows to emulate various other approaches from the literature as well, such as sub-graph isomorphism, maximal common subgraphs, and some edit-distance measures (depending on the cost function).

We believe that due to the employed local-global principle the similarity modeling for workflows becomes feasible. This is because we build upon established similarity modeling approaches that have not just been used in our own POCBR tool CAKE, but in various academic and commercial CBR tools such as myCBR, jColibri, and IAS. However, to proof this, more practical experience with applications involving semantic models of a larger scale is needed.

The developed similarity assessment and retrieval algorithms show a satisfying performance in terms of computation time and retrieval error. We demonstrated scalability to medium sized case bases but an extension to case base with thousands of cases will certainly require additional measures to improve the retrieval performance. Our recent work on scaling semantic workflow retrieval by extending it towards a two-step retrieval approach already led to first promising results [12]. A similar idea for structured cases in general is also investigated by Leake and Kendall-Morwick [30, 28]. Alternatively, a case retrieval

net over the semantic descriptions also seems a suitable approach for this purpose to be investigated in future research. Also, the extension of A*P towards a multi-threaded variant exploiting the parallelization of multi-core CPUs seems promising.

Acknowledgement. This work has its origin in a sabbatical visit of the first author at the Information Sciences Institute in 2010. The authors appreciate the fruitful discussion on the topic of the paper by the various groups' members in Trier and Marina del Rey. This research was funded in part under grant IIS-0948429 from the US National Science Foundation and under grant BE1373/3-1 from the German Research Foundation (DFG).

References

[1] Aamodt, A., Plaza, E., 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Commun. 7, 39–59.

[2] Ankolekar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., McDermott, D., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R., Sycara, K., 2002. DAML-S: web service description for the semantic web, in: The Semantic Web—ISWC 2002, pp. 348–363.

[3] Awad, A., 2007. BPMN-Q: a language to query business processes, in: Reichert, M., Strecker, S., Turowski, K. (Eds.), Enterprise Modelling and Information Systems Architectures - Concepts and Applications , Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'07), St. Goar, Germany, October 8-9, 2007, GI. pp. 115–128.

[4] Becker, J., Bergener, P., Breuker, D., Räckers, M., 2011. On measures of behavioral distance between business processes, in: Proceedings of the 10th International Conference on Wirtschaftsinformatik, pp. 665–674.

[5] Beeri, C., Eyal, A., Kamenkovich, S., Milo, T., 2006. Querying business processes, in: Proceedings of the 32nd international conference on Very large data bases, VLDB Endowment. p. 343–354.

[6] Bergmann, R., 2002. Experience Management - Foundations, Development Methodology, and Internet-Based Applications. volume LNAI 2432. Springer.

[7] Bergmann, R., Althoff, K., Breen, S., Gker, M., Manago, M., Traphner, R., Wess, S., 2003. Developing industrial case-based reasoning applications: The INRECA methodology. volume 1612 of *LNAI*. Springer. 2nd edition.

[8] Bergmann, R., Althoff, K., Minor, M., Reichle, M., Bach, K., 2009. Case-Based reasoning - introduction and recent developments. Künstliche Intelligenz 1/2009, 5–11.

[9] Bergmann, R., Freßmann, A., Maximini, K., Maximini, R., Sauer, T., 2006. Case-based support for collaborative business, in: Roth-Berghofer, T.R., Göker, M.H., Güvenir, A.H. (Eds.), Advances in CBR, Springer. pp. 519–533.

[10] Bergmann, R., Richter, M.M., Schmitt, S., Stahl, A., Vollrath, I., 2001. Utility-oriented matching: A new research direction for case-based reasoning, in: Schmitt, S., Vollrath, I., Reimer, U. (Eds.), 9th German Workshop on Case-Based Reasoning, pp. 264–274.

[11] Bergmann, R., Stahl, A., 1998. Similarity measures for object-oriented case representations, in: Smyth, B., Cunningham, P. (Eds.), Advances in CBR, Springer. pp. 25–37.

[12] Bergmann, R., Minor, M., Islam, S., Schumacher, P., Stromer, A., 2012. Scaling similarity-based retrieval of semantic workflows, in: Lamontagne, L., Recio-Garcia, J. A. (Eds.), ICCBR-Workshop on Process-oriented Case-Based Reasoning, Lyon. pp. 15–24.

[13] Burkhard, H.D., Richter, M.M., 2000. On the notion of similarity in case based reasoning and fuzzy theory, in: Soft computing in case based reasoning, Springer.

[14] Champin, P.A., Solnon, C., 2003. Measuring the similarity of labeled graphs, in: Case-Based Reasoning Research and Development, Springer. p. 1066–1067.

[15] Cheng, W., Rademaker, M., Baets, B.D., Hüllermeier, E., 2010. Predicting partial orders: ranking with abstention, in: Machine Learning and Knowledge Discovery in Databases, p. 215–230.

[16] Chinthaka, E., Ekanayake, J., Leake, D., Plale, B., 2009. CBR based workflow composition assistant, in: World Conference on Services-I,, p. 352–355.

[17] van Der Aalst, W.M., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P., 2003. Workflow patterns. Distributed and parallel databases 14, 5–51.

[18] Dijkman, R., Dumas, M., Garcia-Banuelos, L., 2009. Graph matching algorithms for business process model similarity search, in: Business Process Management, p. 48–63.

[19] Freßmann, A., 2006. Adaptive workflow support for search processes within fire service organisations, in: Reddy, S.M. (Ed.), Proceedings of the fifteenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Computer Society. pp. 291–296.

[20] Gil, Y., Kim, J., Florez, G., Ratnakar, V., González-Calero, P.A., 2009. Workflow matching using semantic metadata, in: Proceedings of the 5th International Conference on Knowledge Capture, p. 121–128.

[21] Gil, Y., Ratnakar, V., Kim, J., González-Calero, P., Groth, P., Moody, J., Deelman, E., 2011. Wings: Intelligent Workflow-Based design of computational experiments. IEEE Intelligent Systems 26, 62–72.

[22] Goderis, A., 2008. Workflow Re-use and Discovery in Bioinformatics. Ph.D. thesis. University of Manchester.

[23] Goderis, A., Li, P., Goble, C., 2008. Workflow discovery: the problem, a case study from e-science and a graph-based solution. International Journal of Web Services Research 5.

[24] Hung, P., Chiu, D., 2004. Developing workflow-based information integration (WII) with exception support in a web services environment, in: System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on, p. 10–pp.

[25] Kapetanakis, S., Petridis, M., Knight, B., Ma, J., Bacon, L., 2010. A case based reasoning approach for the monitoring of business workflows, in: Bichindaritz, I., Montani, S. (Eds.), Case-Based Reasoning. Research and Development, ICCBR 2010, Springer. pp. 390–405.

[26] Kapetanakis, S., Petridis, M., Ma, J., Knight, Brian, Bacon, Liz, 2011. Enhancing similarity measures and context provision for the intelligent monitoring of business processes in CBR-WIMS, in: Proceedings of the ICCBR 2011 Workshops, pp. 87–99.

[27] Kendall-Morwick, J., Leake, D., 2011. A toolkit for representation and retrieval of structured cases, in: Proceedings of the ICCBR 2011 Workshops, pp. 111–120.

[28] Kendall-Morwick, J., Leake, D., 2012. On tuning two-phase retrieval for structured cases, in: Lamontagne, L., Recio-Garcia, J. A. (Eds.), ICCBR-Workshop on Process-oriented Case-Based Reasoning, Lyon. pp. 25–34.

[29] Lawrence, P. (Ed.), 1997. Workflow handbook 1997. John Wiley & Sons, Inc., New York, NY, USA.

[30] Leake, D.B., Kendall-Morwick, J., 2008. Towards Case-Based support for e-Science workflow generation by mining provenance, in: Althoff, K.D., Bergmann, R., Minor, M., Hanft, A. (Eds.), Advances in CBR, pp. 269–283.

[31] Lopez De Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., Watson, I., 2005. Retrieval, reuse, revision and retention in case-based reasoning. Knowl. Eng. Rev. 20, 215–240.

[32] Ludäscher, B., Weske, M., Mcphillips, T., Bowers, S., 2009. Scientific workflows: Business as usual?, in: Proceedings of the 7th International Conference on Business Process Management, Springer-Verlag, Berlin, Heidelberg. p. 31–47.

[33] Lyng, K.M., Hildebrandt, T., Mukkamala, R.R., 2009. From paper based clinical practice guidelines to declarative workflow management, in: Ardagna, D., Mecella, M., Yang, J. (Eds.), Business Process Management Workshops. Springer Berlin Heidelberg, Berlin, Heidelberg. volume 17, pp. 336–347.

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

[35] Maximini, K., Schaaf, M., 2003. The PROGEMM approach for managing clinical processes, in: Kotsis, G., Reddy, S. (Eds.), 1st Workshop on Knowledge Management for Distributed Agile Processes: Models, Techniques, and Infrastructure (KMDAP'03) for the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), IEEE. pp. 332–337.

[36] Minor, M., Bergmann, R., Görg, S., Walter, K., 2010a. Adaptation of cooking instructions following the workflow paradigm, in: Marling, C. (Ed.), ICCBR 2010 Workshop Proceedings.

[37] Minor, M., Bergmann, R., Görg, S., Walter, K., 2010b. Towards Case-Based adaptation of workflows., in: Bichindaritz, I., Montani, S. (Eds.), CBR Research and Development, Springer. pp. 421–435.

[38] Minor, M., Tartakovski, A., Bergmann, R., 2007. Representation and structure-based similarity assessment for agile workflows, in: Weber, R., Richter, M.M. (Eds.), CBR Research and Development, p. 224–238.

[39] Montani, S., Leonardi, G., 2012. Retrieval and clustering for business process monitoring: Results and improvements, in: Agudo, B., Watson, I. (Eds.), Case-Based Reasoning Research and Development. Springer Berlin / Heidelberg. volume 7466 of *Lecture Notes in Computer Science*, pp. 269–283.

[40] Montani, S., Leonardi, G., Lo Vetere, M., 2011. Case retrieval and clustering for business process monitoring, in: Proceedings of the ICCBR 2011 Workshops, pp. 77–86.

[41] Richter, M.M., 2007. Foundations of similarity and utility, in: Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2007), AAAI Press.

[42] Roman, D., Keller, U., Lausen, H., De Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D., 2005. Web service modeling ontology. Applied Ontology 1, 77–106.

[43] Russell, S., Norvig, P., 2010. Artificial intelligence: a modern approach. Prentice hall.

[44] Taylor, I.J., Deelman, E., Gannon, D.B., 2007. Workflows for e-Science. Springer.

[45] Weber, B., Wild, W., Breu, R., 2004. CBRFlow: enabling adaptive workflow management through conversational Case-Based reasoning, in: Funk, P., Gonzlez-Calero, P.A. (Eds.), Advances in CBR, Springer. pp. 434–448.

[46] Witten, I., Frank, E., Trigg, L., Hall, M., Holmes, G., Cunningham, S., 1999. Weka: Practical machine learning tools and techniques with java implementations, in: Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop Future directions for intelligent systems and information sciences.