Adaptation of Scientific Workflows by Means of Process-Oriented Case-Based Reasoning

Christian Zeyen, Lukas Malburg, and Ralph Bergmann

Business Information Systems II, University of Trier, 54286 Trier, Germany
{zeyen,malburgl,bergmann}@uni-trier.de
http://www.wi2.uni-trier.de

Abstract. This paper investigates automatic adaptation of scientific workflows in process-oriented case-based reasoning with the goal of providing modeling assistance. With regard to our previous work on the adaptation of business workflows, we discuss the differences between the workflow types and the implications for transferring the approaches to scientific workflows. An experimental evaluation with RapidMiner workflows demonstrates that the approaches can significantly improve workflows towards a given query while mostly maintaining their executability and semantic correctness.

Keywords: Process-Oriented Case-Based Reasoning · Workflow Adaptation · Scientific Workflows

1 Introduction

In the age of big data, data mining is essential for exploiting the potentials hidden in data. However, defining a suitable data analysis procedure is a very challenging task that requires significant expertise. In e-Science, scientific workflows [23] are an established means for this purpose since they describe at a more abstract level how to address a concrete problem in terms of required data, composition of suitable processing steps, and the selection of good parameter settings. In addition, workflows are a means to document an important step of scientific discovery in a way that reproducibility and reliability of results are improved. Besides their use in natural sciences, their potentials have been recently recognized in the Digital Humanities (DH), where the data mostly consists of text documents which must be processed by natural language and text mining procedures [11,12]. A major obstacle, which currently prevents their wider usage in DH as well as in other fields, is the lack of comprehensive modeling support for scientific workflows that particularly targets the needs of researchers who are not experts in software development and workflow modeling. Scientific Workflow Management Systems (SciWFMs) such as *KEPLER* [14], *Taverna* [22], *RapidMiner* [17], or *WINGS* [7] are very helpful in this respect as they already support the construction and execution of scientific workflows by an integrated visual development environment. Nevertheless, workflow construction remains

a demanding and time-consuming task, in particular for complex data analysis that require combinations of many processing steps [5].

An important line of research aims at supporting the development of scientific workflows by reuse of best-practice workflows [8,5] from a repository [6]. Case-Based Reasoning (CBR) has already been proposed to support the reuse of scientific workflows [4,7,1], however previous work mainly focused on the retrieval of reusable workflows, leaving the necessary adaptation up to the user. In this paper, we now address the adaptation of scientific workflows in a case-based manner. For this purpose, we build upon our previous work in Process-Oriented CBR (POCBR) [18] in which we developed various adaptation methods for business workflows, mostly demonstrated in the field of cooking recipes [19,20,21]. In particular, we investigate the transferability of these approaches to the considerably more complex domain of scientific workflows. This includes the representation of scientific workflows, the learning phase for adaptation knowledge from a case base, as well as the actual adaptation methods using this learned knowledge. Further, we implemented and evaluated the methods for adapting data mining workflows in the SciWFM RapidMiner [17].

The next section reviews related work in the field and analyzes the differences between business and scientific workflows. Section 3 investigates the implications of these differences and presents the results in terms of adaptation approaches that can deal with scientific workflows. Section 4 shows the implementation of the presented approaches for RapidMiner workflows and presents the results of an experimental evaluation. Section 5 concludes with an outlook at future work.

2 Foundations and Related Work

In the following, we briefly survey existing work on modeling support for scientific workflows, we briefly summarize our own work on POCBR which we aim at transferring to scientific workflows, and we discuss the differences between business and scientific workflows.

2.1 Modeling Support for Scientific Workflows

Scientific workflows are executable descriptions of automatable scientific processes such as computational science simulations and data analyses [23]. They consist of a collection of various tasks representing data processing activities such as data import, pre-processing, or modeling together with the parameters used during the execution of these tasks. Also the data items being processed as well as the usage of the data as input or output to the tasks is modeled as part of a scientific workflow. Due to the complexity involved in the creation of scientific workflows, several methods have been proposed to support users in doing so. For instance, an approach by Jannach et al. [9] supports the user with context-sensitive recommendations of single processing steps and parameter settings for the current workflow under construction, based on the analysis of a large workflow repository. Kietz et al. [10] propose a planning approach for

the automatic composition of RapidMiner workflows, correctness-checking, and quick-fixes using an ontology of available computational steps, parameters, and constraints. The SciWFM WINGS [7] also uses planning and semantic reasoning to automatically create workflows based on high-level user requests.

While these approaches require either a fully specified domain model appropriate for planning or a huge number of workflow instances to learn from by induction, CBR has also been used to support workflow development. For instance, Leake and Kendall-Morwick [13] consider execution traces of workflows as cases. The approach extracts tasks from retrieved cases for extending the current workflow under construction. The approach by Chinthaka et al. [4] uses keyword descriptions of inputs and outputs provided by the user for finding the best-matching workflow. If necessary, an adaptation process tries to extend the workflow with single tasks to better fulfill the user's requirements. In our own work, we addressed the retrieval of semantically labeled workflow graphs as a starting point for reuse [16]. In general, previous case-based approaches in the field of scientific workflows mainly focused on retrieval and did not yet address extensive adaptation in a fully automatic manner.

2.2 Process-Oriented Case-Based Reasoning

Our previous work on POCBR already dealt with the adaptation of business workflows. We developed methods for substitutional *adaptation by generalization and specialization* [20] and two methods for structural adaptation, namely *operator-based adaptation* [21] and *workflow streams adaptation* [19]. The methods are currently restricted to business workflows and have been intensively investigated only in the field of cooking recipes. In this paper, we focus on adaptation by generalization and specialization and on workflow stream adaptation.

Adaptation by Generalization and Specialization Using generalized cases [3] is an established approach that allows to represent a set of cases by a single generalized case, leading to a reduced case base and increased coverage. When applied to workflow cases, workflow components such as task and data objects are generalized to a common object that has certain properties that hold for all subsumed objects. During the specialization process, either the original or other available specializations can be inserted to better fulfill a given query.

Adaptation with Workflow Streams Workflow stream adaptation is a compositional adaptation approach [24] which decomposes the workflow cases of the case base into meaningful sub-workflows (referred to as workflow stream) that are stored as adaptation knowledge. A retrieved workflow is adapted with respect to a query by incrementally replacing its own workflow streams with more suitable streams from the adaptation knowledge.

Learning Adaptation Knowledge Since the presented adaptation methods require a significant amount of domain-specific adaptation knowledge, the developed methods automatically learn required knowledge (generalized workflows and workflow streams) from the workflow repository, i.e., the case base. Hence, we distinguish between a learning phase of adaptation knowledge and a problem solving phase in which for a given query the best matching workflow is

adapted such that it matches the particular problem scenario at best. By applying the learned adaptation knowledge between different cases, a larger solution space is covered.

2.3 Differences Between Business and Scientific Workflows

There are significant differences between business workflows and scientific workflows, which prevent the direct application of the developed adaptation methods.

Business workflows aim at automating organizational processes, which are mainly executed by humans, supported by certain resources including application programs. While the goal of a business workflow is already determined before the workflow is executed, the goal of a scientific workflow is to validate hypotheses of a researcher based on available data. These scientific goals are experimental and exploratory and thus vary more frequently. While business workflows are executed under the control of the involved humans, scientific workflows are executed fully automatically by a computer [15]. Consequently, they depend more strongly on proper parameter settings.

Control-flow and data-flow orientation are discussed as main differences between business and scientific workflows [15,1]. In business workflows, control-flow patterns such as *AND*, *XOR*, and *LOOPS* are commonly used. Data-flow is modeled separately or implicitly due to its subordinate significance [15]. In scientific workflows, data-flow is of primary importance since the control-flow, respectively the execution order of the computational steps, results implicitly from the given data-flow. A computational step can only be executed after the required data has been generated by the previous steps, hence the workflow is data-driven. However, some SciWFMs also enable to explicitly define control-flow between computational steps or to use control-flow patterns.

Many SciWFMs such as *KEPLER* [14], *Taverna* [22], or *RapidMiner* [17] use certain interfaces (referred to as ports) for restricting the data-flow between computational steps to ensure that only valid data is exchanged. Port connections are not used in business workflows, which is thus a further significant difference.

3 Adaptation of Scientific Workflows

This section outlines the representation of scientific workflows as semantic graphs before the substitutional and structural adaptation approaches are described. We keep those descriptions on an informal level as the full algorithmic details[1] would exceed the space limitation of this paper.

3.1 Graph Representation and Semantic Similarity

To represent workflows, we use semantic graphs [1]. Figure 1 depicts such a graph of a scientific workflow from the data mining domain. In a semantic graph,

[1] Lukas Malburg formally described the approaches in his master thesis "Adaptation of Scientific Workflows by Means of POCBR" submitted 2019 at Trier University.

semantic descriptions and specific types are assigned to each node and edge. Semantic descriptions are based on a semantic metadata language. For scientific workflows, we use object-oriented descriptions to represent heterogeneous parameters and values of computational steps. The computational steps of a scientific workflow are represented as task nodes in the graph. Data nodes between task nodes represent the data that is exchanged between computational steps. To represent port connections, corresponding information is added to the semantic descriptions. In the example graph, the semantic descriptions of data nodes comprise information about the labels of connected input and output ports. For example, it is stated that data object *Table Data* is produced by the task *Import Data* at the output port *table data* and that the same data object is consumed by the task *Discretize* at the input port *table data*.

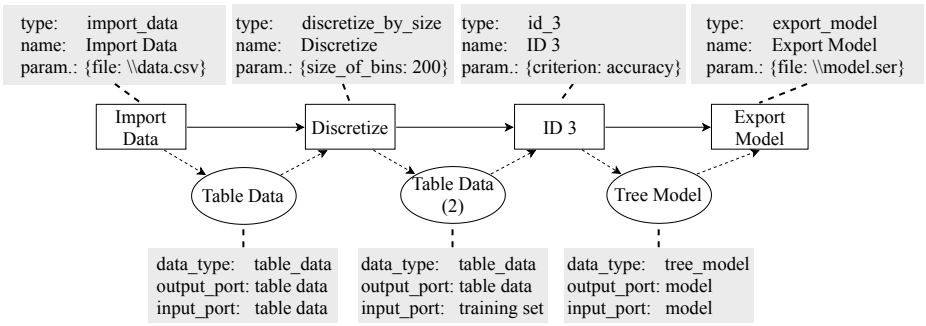


Fig. 1. Graph Representation of a Scientific Workflow

In a scientific workflow graph, the data nodes reflect the state changes of processed data and thus have to be distinct. For instance, the task *Discretize* produces an output of the same data type but in an altered state. Hence, data nodes *Table Data* and *Table Data (2)* are represented as distinct nodes. In business workflows (cf. the cooking workflows used in our previous work [19,20]), each data node is unique, too, but state changes of data are typically not represented. Due to this difference, the usage of data-flow edges between data nodes and task nodes differs in the graph representation. While a data node can be linked with various task nodes as input or output in a business workflow graph, we restrict the connection of a data node such that it can only be connected as output of one task node and as input of another task node in a scientific workflow graph.

Task nodes are also semantically enriched by semantic descriptions that contain, for example, the type of the computational step, the name, and defined parameters. Even though control-flow is secondary for scientific workflows, we explicitly represent the control-flow between task nodes by selecting one valid execution order of tasks. By this means, we take into account SciWFMs such as RapidMiner that allow for defining control-flow.

Analogous to previous work (cf. [19,20]), we use the semantic similarity measure by Bergmann and Gil [1] to assess the similarity between two graphs. The measure uses subgraph matching and applies heuristic search to find the best possible, injective, partial mapping m between the nodes and edges of the query workflow QW and those of the case workflow CW:

$$\mathrm{sim}(QW, CW) = \max\{\mathrm{sim}_m(QW, CW) \mid \text{mapping } m\} \tag{1}$$

During the search, each mapping of two nodes or edges is rated by local similarity functions $\mathrm{sim}_m \rightarrow [0, 1]$. Following the local-global principle, similarities between a pair of mapped nodes or edges is assessed by comparing the attribute values of their semantic descriptions with each other. Such similarities are then aggregated to similarities of mapped nodes or edges, which, in turn, are aggregated to the global similarity value of two graphs.

3.2 Substitutional Adaptation by Generalization and Specialization

The adaptation by generalization and specialization is used to substitute task nodes in a scientific workflow graph. In the learning phase, the workflows in the case base are generalized. In the problem solving phase, a retrieval is performed in the generalized case base for the best-matching generalized workflow. Subsequently, the generalized workflow is specialized in regards to the query.

Generalization The generalization requires ontological information about the tasks. In particular, a taxonomy of all types of tasks is required. In such a hierarchy, the generalization considers the types along the path from a concrete type (i.e., a leaf node) towards the root as possible generalizations [20]. Figure 2 depicts an excerpt of an exemplary task taxonomy. In the taxonomy,

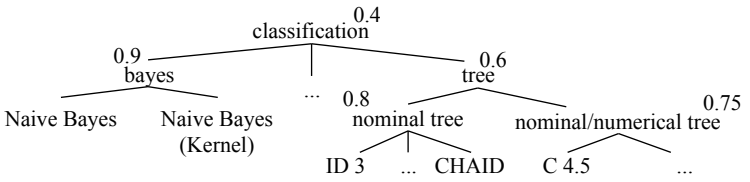


Fig. 2. Excerpt of a Task Taxonomy

the generalized class *bayes* subsumes the concrete tasks *Naive Bayes* and *Naive Bayes (Kernel)* and has assigned a semantic similarity value of 0.9 that holds for all subsumed tasks, i.e., for both naive bayes implementations. The values were determined manually considering the meta data of tasks. In contrast to business workflows, the generalization of scientific workflows primarily relates to tasks since data types depend on tasks and thus cannot be adapted independently. In order to generalize a task node in a workflow graph, several conditions must be satisfied. In previous work [20], two similarity thresholds have been proposed for

business workflows to assess whether task nodes are suitable for generalization. The first similarity threshold $\triangle_W \in [0, 1]$ prevents workflows from being generalized that are considered too heterogeneous. A second threshold is defined for a measure that takes into account the taxonomic structure of the tasks to prevent over-generalization. For scientific workflows, this parameter is not required, since port connections of tasks are considered as constraints instead. Two tasks from different workflows are considered to be generalizable, if they have equal port connections, i.e., if the specifications of connected ports are identical. Furthermore, it is required that the generalized task is able to consume and produce the same data items. For each inner node (representing a generalized task type) in the taxonomy, we determine the common port specifications of the subsumed tasks and assign them to the inner node. A task is incrementally generalized until no more common port specifications exist. The constraint of common port connections ensures that each specific task is able to consume and produce the same data as its siblings in the taxonomy. Thus, in the specialization phase, each specific child of a generalized task can be selected for substitution and it is ensured that no structural changes (e.g., adaptation of the data-flow) to the workflow are required.

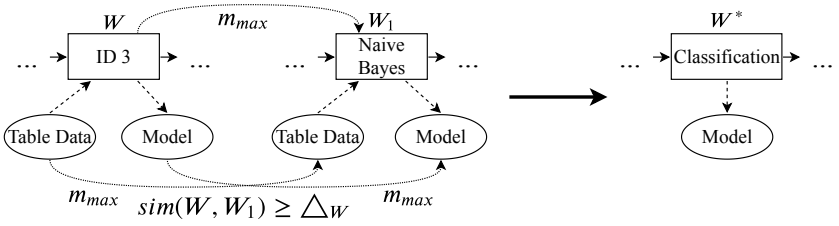


Fig. 3. Exemplary Generalization of Tasks

Figure 3 illustrates an exemplary generalization process. In the given example, two similar workflow graphs W and W_1 are compared with a similarity $\text{sim}(W, W_1) \geq \triangle_W$. The task node *ID 3* of workflow W is mapped to the task node *Naive Bayes* of workflow W_1. Both tasks consume and produce the same data. Assuming that the tasks have the same port connections and that the generalized task type *Classification* is a common ancestor of both tasks (see Figure 2) that is able to consume and produce the corresponding data, the task *ID 3* can be generalized to *Classification*.

Specialization The specialization process is similar to the process applied to business workflows (cf. [20]). For each generalized task in a retrieved generalized workflow, the most suitable specializations are inserted with respect to the given query. If a generalized task is used in the query workflow, an arbitrary specific task is selected during the specialization of business workflows. However, to ensure the executability of scientific workflows, already known specializations are preferred over unknown specializations. Hence, during generalization all original

tasks are stored for each generalized task. If a known specialization exists, it is chosen as specialization. Otherwise, an arbitrary specialization is selected. In this event, the task is pre-initialized with known default parameter settings.

3.3 Structural Adaptation with Workflow Streams

The adaptation with workflow streams exploits the structure of workflows to define the borders of substitutable and meaningful sub-workflows that we call workflow streams. For this purpose, the approach poses the requirement of *block-orientation* (cf. [19]) on the representation of workflow graphs. In a nutshell, block-orientation restricts the control-flow in a graph in such a way that only a single start and end node is allowed and that all task nodes must be connected by control-flow edges. For workflow representations without an explicit definition of the control-flow, which is often the case for scientific workflows, the control-flow must be made explicit to be block-oriented.

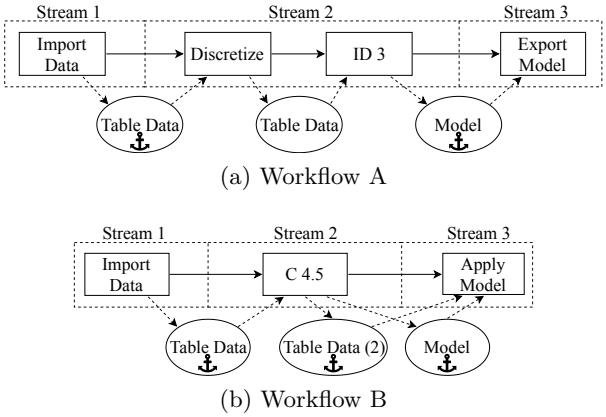
A workflow stream is considered as a sub-workflow that produces a partial output of a workflow. A task that produces such a data object is referred to as *producer* (or creator). Due to the different representation of data objects in scientific workflow graphs (cf. Section 3.1), the definition of a producer differs from our previous approach. In business workflow graphs, types of data nodes are always considered different. Since data objects are stateful in scientific workflows, several data nodes may exist with an equal data type. Thus, producers are identified by looking at the input and output data types. A task is considered as producer, if it produces at least one data object of a type that is different from all of its input data types.

Workflow Stream Partitioning A block-oriented scientific workflow can be partitioned into workflow streams similarly to a business workflow. Based on the identified producers $P \subseteq T$ from all tasks T in the workflow, the workflow is partitioned into a set of workflow streams such that each task $t \in T$ is exactly assigned to one stream. A workflow stream S is constructed for each producer $p \in P$. The producer p constitutes the end task node in the control-flow of the stream S. A task $t \in T \setminus P$ is part of the stream S,

- if t is a predecessor of p in the control-flow,
- if t is not already part of another stream,
- and if t is connected to an equal data node the producer p or another task of stream S is connected to (cf. definitions of data-flow connectedness in [19]).

In such a partitioning, each task is exactly assigned to one workflow stream. Figure 4 illustrates the partitioning of two scientific workflows. All workflow streams are marked with dashed rectangles. The producers in the example workflow A are *Import Data* and *ID 3*. The example demonstrates that streams do not only produce partial outputs, but also consume data (see stream 3).

Workflow Stream Substitution For the problem-solving phase, we assume that the available workflows in the case base have been decomposed into workflow streams in a learning phase beforehand. A workflow from the case base can be



(a) Workflow A

(b) Workflow B

Fig. 4. Partitioning of Scientific Workflows

adapted w.r.t. a given query workflow by replacing its workflow streams with other available streams that better fulfill the requirements of the query. For this purpose, the workflow to be adapted is decomposed into streams and for each stream a search for the best substitute stream candidate is performed. This search process applies the semantic graph similarity outlined in Section 3.1 to find the best workflow streams with respect to a given query.

Two streams are regarded as substitutable, if they consume and produce the same data. More precisely, a stream S_1 is substitutable by a stream S_2, if all the data objects in S_1 that are consumed or produced by a task of another stream are identical to those data objects of S_2. Such data nodes are also referred to as *data anchors* and the corresponding task is named *consumer* or *producer anchor*, respectively. In Figure 4, data anchors are marked with ⚓ symbols.

Due to the availability of port connections in scientific workflows, a further condition is proposed to define substitutable streams. If the set of data anchors of two streams S_1 and S_2 is not identical, it is checked for each unmatched data anchor of stream S_1 whether the other stream S_2 contains a consumer or producer anchor with an unused port that is suited to consume or produce the data anchor of S_1. For instance, in the example given in Figure 4, Stream 2 of Workflow B is substitutable by Stream 2 of Workflow A, assuming that both tasks *C 4.5* and *ID 3* have equal output port specifications. In this event, the unused port of *ID 3* is used to produce the data object *Table Data (2)*.

3.4 Combined Adaptation

Both adaptation approaches can also be used in combination. By this means, the coverage of the solution space can be increased, which has been investigated in previous work [2]. For the combined adaptation, the case base is first generalized to learn generalized workflow streams. Then, retrieval is performed and the best-matching generalized workflows are adapted. First, structural adaptation

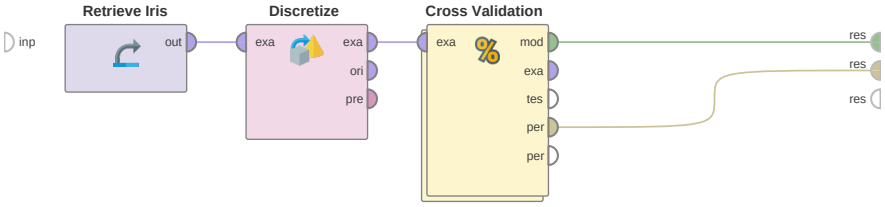
substitutes generalized workflow streams that best match the given query. Subsequently, substitutional adaptation specializes the adapted workflow to produce concrete and executable workflows.

4 Implementation and Experimental Evaluation

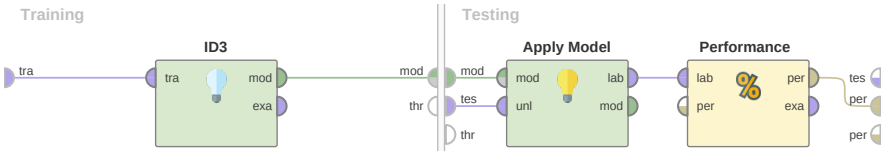
We fully implemented the described approaches in the POCBR component of the CAKE framework[2] such that it can adapt RapidMiner workflows. The following section outlines the implementation with a particular focus on the semantic graph representation as prerequisite of adaptation. Subsequently, we present the evaluation setup and discuss the results.

4.1 Implementation of the RapidMiner Domain

We developed a plugin for the RapidMiner environment [17] that allows for automatically exporting workflows to XML and extracting meta data about available workflow components. Essentially, the knowledge model is based on these meta data that comprise various information about each available task such as textual descriptions, parameters, value ranges, default settings, input and output ports, and data types. Figure 5 illustrates a data mining workflow that reads



(a) Total View of the Workflow



(b) Nested Sub-workflows of the Cross Validation Task

Fig. 5. Screenshot of a RapidMiner Workflow

and discretizes a data set, learns an ID3 decision tree, and performs a cross validation to measure the model performance. Some tasks (in the following referred to as complex tasks) such as the Cross Validation enclose further tasks as sub-workflows. Figure 6 shows the semantic graph representation of the over-

[2] See http://procake.uni-trier.de

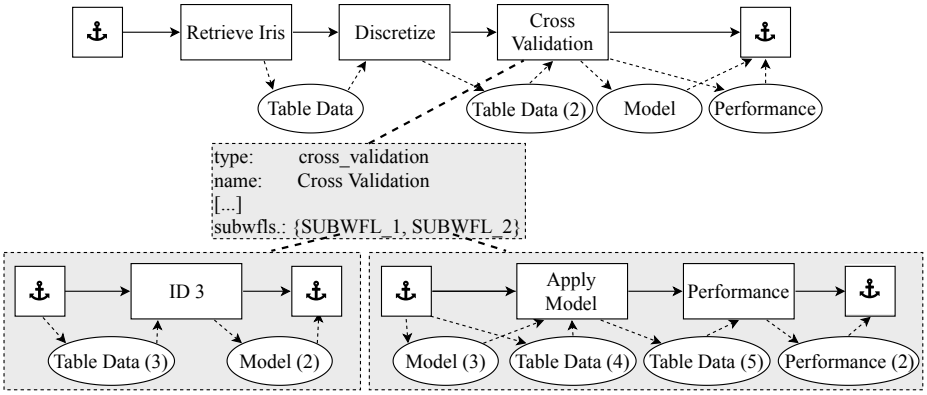


Fig. 6. Semantic Graph of a RapidMiner Workflow

all workflow, which is similar to the graph depicted in Figure 1. It should be noted that only the semantic description of the complex task is shown in the figure. Sub-workflows of a complex task are represented as semantic graphs that are referenced in the corresponding semantic description. In the given example, both graphs are listed as references in the *subworkflows* attribute of the semantic description. Hence, when computing the similarity between two complex tasks, referenced sub-workflows are also compared according to their order in the *subworkflows* attribute using new instances of the graph similarity measure. By this means, they contribute to the semantic similarity of complex tasks. Moreover, during the similarity computation of the higher-level workflow graph, workflow components can also be mapped between the higher-level graph and the sub-workflow graphs. Consequently, a retrieval with a query that does not contain sub-workflows can still find similar workflows that contain similar components nested in sub-workflow structures. To represent data-flow connections to and from workflow or sub-workflow input and output ports, auxiliary tasks referred to as *IO anchors* are added to the graphs (see ⚓ symbols in Figure 6).

The presented adaptation approaches can be applied for this graph representation. An exception are io anchors, which are not adapted by the methods to ensure that existing data-flow connections are retained. Regarding the requirement of block-orientation for the adaptation with workflow streams, sub-workflows can be adapted separately, since IO anchors ensure that their interfaces to other workflows are considered.

4.2 Hypotheses and Evaluation Setup

To evaluate the adaptation methods, we perform adaptation experiments with executable RapidMiner workflows and constructed queries. With respect to the average query fulfillments, we investigate the following hypotheses:

H1 Each adaptation method provides at least as good results as the sole retrieval.

H2 The structural adaptation method outperforms the substitutional adaptation method.
H3 The combined adaptation method provides better results as both, the structural and the substitutional adaptation method.
H4 The adaptation methods produce executable and semantically correct workflows.

As evaluation criteria, we measure the *query fulfillment*, which is determined by the semantic similarity between the query and the adapted workflow graphs (see Section 3.1). Furthermore, we assess the *quality*, which is determined by executability and semantic correctness of the adapted workflows.

For the experiments, we created a case base of 20 data mining workflows as follows: First, we selected three sub-trees of the task taxonomy that subsume classification and clustering tasks (e.g., ID3 or k-Means), validation tasks (e.g., cross or split validation), and performance measuring tasks (e.g., accuracy or cluster distance). Subsequently, we randomly selected one concrete task from each of the sub-trees and constructed an executable workflow containing these tasks by adding required data processing tasks. All workflows are created for the same data set to allow for a broader applicability of adaptation knowledge. Figure 5 depicts one of the created workflows. Similarly, we created 10 queries, which are different from one another and from the workflows in the case base, by randomly selecting one task from each of the three sub-trees, including generalized tasks (i.e., inner tree nodes). Such a query is a partial, non-executable workflow whose tasks are not connected with each other. Here, additionally required data processing tasks are not included.

For each query, a retrieval of the k top-ranked workflows is performed in the case base. Workflows with an identical similarity value are considered equally ranked. Subsequently, adaptation is performed on the top-k workflows and the *query fulfillment* is measured. The workflows with the highest query fulfillment are selected for further examination. We checked the *syntactic correctness* by converting the graph into the XML format, the *executability* by importing (the XML) and executing the workflow in RapidMiner, and the *semantic correctness* by manual examination after successful execution.

As baseline for the query fulfillment, a *retrieval without adaptation* is conducted. For the *structural adaptation* approach, retrieval is performed and the k top-ranked workflows are adapted w.r.t. the query. For the *substitutional adaptation* approach, the case base is generalized first, a retrieval is performed with the generalized workflows, and the k top-ranked workflows are specialized. Due to a rather narrow scope of the workflows, the threshold $\triangle W = 0$ is used for the generalization. In a third experiment, the *combined adaptation* is evaluated.

4.3 Results

Table 1 summarizes the measured query fulfillments for plain retrieval, i.e., the baseline, (row "w/o") and for the performed adaptations. An empty space in the table indicates that the measured value is equal to the baseline. For $k > 1$

the highest value is taken, since adaptability among the workflows may vary. A cross mark beside a value indicates that none of the k top-ranked workflows is executable or semantically correct. In each of these events, the error is caused by missing pre-processing tasks for converting the data set into a format required by the requested classification or clustering task (e.g., discretization). We found no case, where an adapted workflow was executable but not semantically correct. The results of structural adaptation show that adaptation of the first

Table 1. Maximum Query Fulfillments of k top-ranked adapted Workflows

		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	avg
w/o		0.82	0.60	0.85	0.85	0.91	0.81	0.78	0.85	0.71	0.72	0.79
	$k = 1$			1.00			1.00 ✗	1.00			0.86 ✗	0.86
struct.	$k = 2$	1.00		1.00	1.00		1.00 ✗	1.00		0.91	0.86 ✗	0.91
	$k \geq 3$	1.00	0.69	1.00	1.00		1.00 ✗	1.00		0.91	0.86 ✗	0.92
subst.	$k \geq 1$	1.00	0.68		1.00		1.00 ✗	1.00		0.91	0.91	0.91
comb.	$k \geq 1$	1.00	0.69	1.00	1.00		1.00 ✗	1.00		0.91	1.00 ✗	0.94

ranked workflows by retrieval does not necessarily lead to the highest possible similarity, which is only reached with $k \geq 3$. In the experiments with the substitutional adaptation, the generalized workflows ranked highest by retrieval are also best suited for adaptation regarding each query. The query fulfillments do not increase for higher k values. Both adaptation methods outperform the sole retrieval, which confirms Hypothesis H1. No significant differences in the results of strutural and substituional adaptation could be observed in the experiments. Thus, Hypothesis H2 cannot be confirmed. For $k = 1$, the combined approach achieved the best results on average, followed by the substitutional and structural approaches. Hence, Hypothesis H3 is confirmed. However, comparing these similarities with those obtained without adaptation, all approaches yield to significantly ($p < 0.05$) higher values on average.

The results also show some differences between the queries. In the events, where adaptation does not exceed retrieval ($Q5$ and $Q8$) or where adapted workflows are not executable ($Q6$ and $Q10$), queries contain combinations of tasks for which no or at least no applicable adaptation knowledge is available. For example, in $Q6$, an ID3 decision tree is requested that requires discretized data, but a discretization task is not part of the query. All the adapted workflows contain the requested tasks but no discretization task since it is not requested. Consequently, Hypothesis H4 is only partially confirmed.

5 Conclusions

This work investigates the application of adaptation in POCBR for the complex domain of scientific workflows. In a first step, we determine differences be-

tween business and scientific workflows. These differences affect the adaptation methods that are already used to adapt business workflows. Thus, we present a modified approach for structural adaptation with *workflow streams* and for substitutional adaptation by *generalization and specialization*. We implement both adaptation methods in the CAKE framework for adapting RapidMiner workflows. Both adaptation methods show promising results in the experimental evaluation as they are able to significantly increase the query fulfillment.

Due to the differences between business and scientific workflows and the more complex domain of scientific workflows, further advancements in the adaptation approaches are necessary to improve executability and semantic correctness of adapted workflows. At present, adaptation only ensures the syntactic correctness and does not consider or adapt parameter settings of tasks. For future work, several ideas for improving the adaptation methods exist. The structural adaptation can be improved by adding an insert capability and by including finer-grained adaptation components such as those presented in our operator-based adaptation approach. Moreover, it can be investigated how dependencies of tasks can be determined for a given data set. For this purpose, the coverage of adaptation can be computed by applying the entire adaptation knowledge to the workflows in the case base. The resulting sets of executable and non-executable workflows can then be analyzed regarding the dependencies of tasks to derive, which adaptation components are compatible with each other.

Further, we plan to expand the evaluation to more complex, user-generated workflows and we will integrate the adaptation methods within our plugin for the RapidMiner environment in order to support their use in a convenient way. By this means, a further step towards the interactive adaptation of workflows for providing modeling assistance can be made.

Acknowledgments. This work is partly funded by the German Federal Ministry of Education and Research (BMBF, No. 01UG1606) and the German Research Foundation (DFG, No. BE 1373/3-3).

References

1. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. Information Systems **40**, 115–127 (2014)
2. Bergmann, R., Minor, M., Müller, G., Schumacher, P.: Project EVER: Extraction and Processing of Procedural Experience Knowledge in Workflows. In: Proceedings of ICCBR 2017 Workshops. vol. 2028, pp. 137–146. CEUR-WS.org (2017)
3. Bergmann, R., Vollrath, I.: Generalized Cases: Representation and Steps Towards Efficient Similarity Assessment. In: Proceedings of the 23rd Annual German Conference on Artificial Intelligence. LNCS, vol. 1701, pp. 195–206. Springer (1999)
4. Chinthaka, E., Ekanayake, J., Leake, D.B., Plale, B.: CBR Based Workflow Composition Assistant. In: 2009 IEEE Congress on Services, Part I, SERVICES I 2009. pp. 352–355. IEEE Computer Society (2009)
5. Cohen-Boulakia, S., Leser, U.: Search, Adapt, and Reuse: The Future of Scientific Workflows. SIGMOD Record **40**(2), 6–16 (2011)

6. De Roure, D., Goble, C., Bhagat, J., Cruickshank, D., Goderis, A., Michaelides, D., Newman, D.: myExperiment: Defining the social virtual research environment. In: IEEE Fourth International Conference on eScience. pp. 182–189. IEEE (2008)
7. Gil, Y., Ratnakar, V., Kim, J., González-Calero, P.A., Groth, P.T., Moody, J., Deelman, E.: Wings: Intelligent Workflow-Based Design of Computational Experiments. IEEE Intelligent Systems **26**(1), 62–72 (2011)
8. Goderis, A.: Workflow Re-use and Discovery in Bioinformatics. Ph.D. thesis, University of Manchester (2008)
9. Jannach, D., Jugovac, M., Lerche, L.: Supporting the Design of Machine Learning Workflows with a Recommendation System. TiiS **6**(1), 8:1–8:35 (2016)
10. Kietz, J.U., Serban, F., Fischer, S., Bernstein, A.: "Semantics Inside!" But Let's Not Tell the Data Miners: Intelligent Support for Data Mining. In: Proc. of the 11th Int. Semantic Web Conf. LNCS, vol. 8465, pp. 706–720. Springer (2014)
11. Kuhn, J., Reiter, N.: A Plea for a Method-Driven Agenda in the Digital Humanities. In: Book of Abstracts of DH 2015 (2015)
12. Kuras, C., Eckar, T.: Prozessmodellierung mittels BPMN in Forschungsinfrastrukturen der Digital Humanities. In: INFORMATIK 2017. pp. 1101–1112. GI (2017)
13. Leake, D.B., Kendall-Morwick, J.: Towards Case-Based Support for e-Science Workflow Generation by Mining Provenance. In: Advances in Case-Based Reasoning, ECCBR 2008. Proceedings. LNCS, vol. 5239, pp. 269–283. Springer (2008)
14. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M.B., Lee, E.A., Tao, J., Zhao, Y.: Scientific Workflow Management and the KEPLER System. Concurrency and Computation: Practice and Experience **18**(10), 1039–1065 (2006)
15. Ludäscher, B., Weske, M., McPhillips, T.M., Bowers, S.: Scientific Workflows: Business as Usual? In: Business Process Management, 7th International Conference, BPM 2009, Ulm. Proceedings. pp. 31–47. LNCS, Springer (2009)
16. Malburg, L., Münster, N., Zeyen, C., Bergmann, R.: Query Model and Similarity-Based Retrieval for Workflow Reuse in the Digital Humanities. In: Proceedings of LWDA 2018, Mannheim. vol. 2191, pp. 251–262. CEUR-WS.org (2018)
17. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid prototyping for complex data mining tasks. In: Proceedings of the 12th ACM SIGKDD Int. Conf. on Knowl. Discovery and Data Mining, 2006. pp. 935–940. ACM (2006)
18. Minor, M., Montani, S., Recio-García, J.A.: Process-oriented case-based reasoning. Information Systems **40**, 103 – 105 (2014)
19. Müller, G., Bergmann, R.: Workflow Streams: A Means for Compositional Adaptation in Process-Oriented CBR. In: Case-Based Reasoning Research and Development, ICCBR 2014. LNCS, vol. 8765, pp. 315–329. Springer (2014)
20. Müller, G., Bergmann, R.: Generalization of Workflows in Process-Oriented Case-Based Reasoning. In: Proc. of FLAIRS 2015. pp. 391–396. AAAI Press (2015)
21. Müller, G., Bergmann, R.: Learning and Applying Adaptation Operators in Process-Oriented Case-Based Reasoning. In: Case-Based Reasoning Research and Development, ICCBR 2015. LNCS, vol. 9343, pp. 259–274. Springer (2015)
22. Oinn, T., et al.: Taverna/myGrid: Aligning a Workflow System with the Life Sciences Community. In: Workflows for e-Science, Scientific Workflows for Grids. Springer UK (2006)
23. Taylor, I.J., Gannon, D.B., Shields, M. (eds.): Workflows for e-Science: Scientific Workflows for Grids. Springer London (2010)
24. Wilke, W., Bergmann, R.: Techniques and knowledge used for adaptation during case-based problem solving. In: Tasks and Methods in Applied Artificial Intelligence. pp. 497–506. Springer Berlin Heidelberg (1998)