Keyword-Based Search of Workflow Fragments and Their Composition

Khalid Belhajjame[1(✉)], Daniela Grigori[1], Mariem Harmassi[1,2], and Manel Ben Yahia[1]

[1] Université Paris-Dauphine, PSL Research University, CNRS, UMR [7243], LAMSADE, 75016 Paris, France
{khalid.belhajjame,daniela.grigori,mariem.harmassi, manel.benyahia}@dauphine.fr

[2] L3i Lab, Université de La Rochelle, La Rochelle, France
mariem.harmassi@univ-lr.fr

Abstract. Workflow specification, in science as in business, can be a difficult task, since it requires a deep knowledge of the domain to be able to model the chaining of the steps that compose the process of interest, as well as awareness of the computational tools, e.g., services, that can be utilized to enact such steps. To assist designers in this task, we investigate in this paper a methodology that consists in exploiting existing workflow specifications that are stored and shared in repositories, to identify workflow fragments that can be re-utilized and re-purposed by designers when specifying new workflows. Specifically, we present a method for identifying fragments that are frequently used across workflows in existing repositories, and therefore are likely to incarnate patterns that can be reused in new workflows. We present a keyword-based search method for identifying the fragments that are relevant for the needs of a given workflow designer. We go on to present an algorithm for composing the retrieved fragments with the initial (incomplete) workflow that the user designed, based on compatibility rules that we identified, and showcase how the algorithm operates using an example from eScience.

1 Introduction

Workflows are popular means for specifying and enacting processes in business as in science. For example, they are used in modern sciences to specify and enact in-silico experiments, thereby allowing scientists to gain better understanding of the phenomenon or hypothesis they are investigating. The design of scientific workflows can however be a difficult task as it requires a deep knowledge of the domain as well as awareness of the programs and services available for implementing the workflow steps. To overcome this obstacle and facilitate the design of workflows, many workflows repositories have emerged, e.g., myExperiment [1],

The research reported on this paper was supported by the french research agency (ANR-14-CE23-0006). M. Harmassi and M. Ben Yahia contributed to this work during their masters internship at the Lamsade Laboratory.

N.T. Nguyen et al. (Eds.): TCCI XXVI, LNCS 10190, pp. 67–90, 2017.
DOI: 10.1007/978-3-319-59268-8_4

Crowdlabs [2] and Galaxy [3] to share, publish and enable reuse of workflows. For example, De Roure *et al.* [1] pointed out the advantages of sharing and reusing workflows as a solution to face the difficulty and cost of design.

Sharing and publishing workflows is however not sufficient to enable their reuse. Over the past years, an important number of workflows has been shared by scientists in several domains on the myExperiment workflow repository. However, their users face difficulties when it comes to exploring and querying workflows. Indeed, users still have to go through published workflows to identify those that are relevant for their needs. The situation is exacerbated by the fact that the number of workflows hosted by workflow repositories is rapidly increasing. To overcome this problem, mining techniques can be utilized to automatically analyze the workflows in the repository with the objective to provide templates that assist users in the design of their own workflows, thereby allowing them to take advantage of a knowledge-asset gained and verified by their peers.

Several works have been proposed in the literature for mining workflows (see, e.g. [4–6]). Unlike these proposals, our objective is not to propose yet another mining algorithm. Instead, we investigate the graph representations that can be used to encode workflow specifications into graphs before they are examined by existing graph mining algorithms. We are particularly interested in sub-graph mining techniques that find commonalities among fragments of workflows. Indeed, fragments that are common to multiple workflows are likely to be patterns that can be useful for designers when specifying new workflows. In elaborating possible representations, we take into consideration the cost in terms of time that the graph mining algorithm spends given a workflow representation, and the impact of the representation on the quality of the mining algorithm results.

As well as mining frequent workflows fragments, we investigate the problem of exploring them by designers using keyword search. In doing so, we augment traditional TF-IDF with semantic capabilities that take into consideration synonym relation between the keywords used by users in their query and the terms used to label the activities that compose workflow fragments. Furthermore, we elaborate an algorithm for assisting designers in the composition of the retrieved fragments with the initial (incomplete) workflows that they specified.

Accordingly, the contributions of this paper are as follows.

– We elaborate representation models for encoding workflows in the form of graphs that can be used as input to sub-graph mining algorithms, and systematically evaluate the effectiveness of such representations through an empirical evaluation (in Sect. 4).
– We present a keyword-based search method for identifying relevant frequent graphs (in Sect. 5).
– We present an algorithm for assisting designers in the composition of the workflow fragments with their (incomplete) workflow specification (in Sect. 6), and showcase how it operates using an example from eScience (in Sect. 7).

Furthermore, we present the overall approach (in Sect. 2), review and compare existing proposals to ours (in Sect. 3). Finally, we conclude the paper underlining the main contributions and discussing future research directions (in Sect. 8).

The work reported in this paper is an extended version of the work presented in [7]. In [7], we have investigated the representation models suitable for mining workflow fragments (Sect. 4). In the extended version reported on in this paper, we make the following new contributions. We (i) investigate keyword search of workflow fragments (Sect. 5), (ii) study the problem of workflow fragment composition (Sect. 6), (iii) show how the composition operates using an example (Sect. 7). Furthermore, we weave the three pieces of mining, keyword search, and composition of workflow fragments within a global method (Sect. 2), and extend related work analysis (Sect. 3).

2 Approach Overview

Designing a workflow is a time consuming and sometimes expensive task. The designer needs to be knowledgeable of the tools (services) that are available to implement the steps in the process she desires. Furthermore, she needs to know how such services can (or should) be connected together taking into consideration, amongst other criteria, the types and semantic domains of their input and output parameters. There has been a good body of work in the literature on workflow discovery, see e.g., [8–10]. The typical approach requires the user to specify a query in the form of a workflow that is compared with a collection of workflows. The workflows that are retrieved are then ranked taking into consideration, amongst other things, the structural similarity between the workflow issued by the user and the workflows retrieved. In our work, we focus on a problem that is different and that received little attention in the literature. Specifically, we position ourselves in a context where a designer is specifying her workflow and needs assistance to design parts of her workflow, e.g., because she does not know the services that are available and that can be used to enact the steps within the part in question. In certain situations, the designer may know the services that can be used for such purpose, but would still like to acquire knowledge about the best way/practice for connecting such services together. The solution we describe in this paper has been elaborated with the needs of such designers in mind. It can be used to assist them finding an existing fragment that can be used to complete the workflow being designed. Furthermore, we provide the user with suggestions on the way such fragments can be composed with the initial workflow.

Figure 1 illustrates our approach using two processes. Figure 1(a) illustrates the process that is enacted offline to build a repository of workflow fragments. Specifically, given a workflow repository, e.g., the myExperiment repository [1], the labels used to name the activities in the workflow are homogenized. Indeed, different designers are likely to use different labels to name activities that perform the same task. For this purpose, we use existing state of the art techniques, which consist in using shared vocabularies (dictionaries) to rename the activities of the workflow. Once the labeling of the workflows in the repository is homogenized,

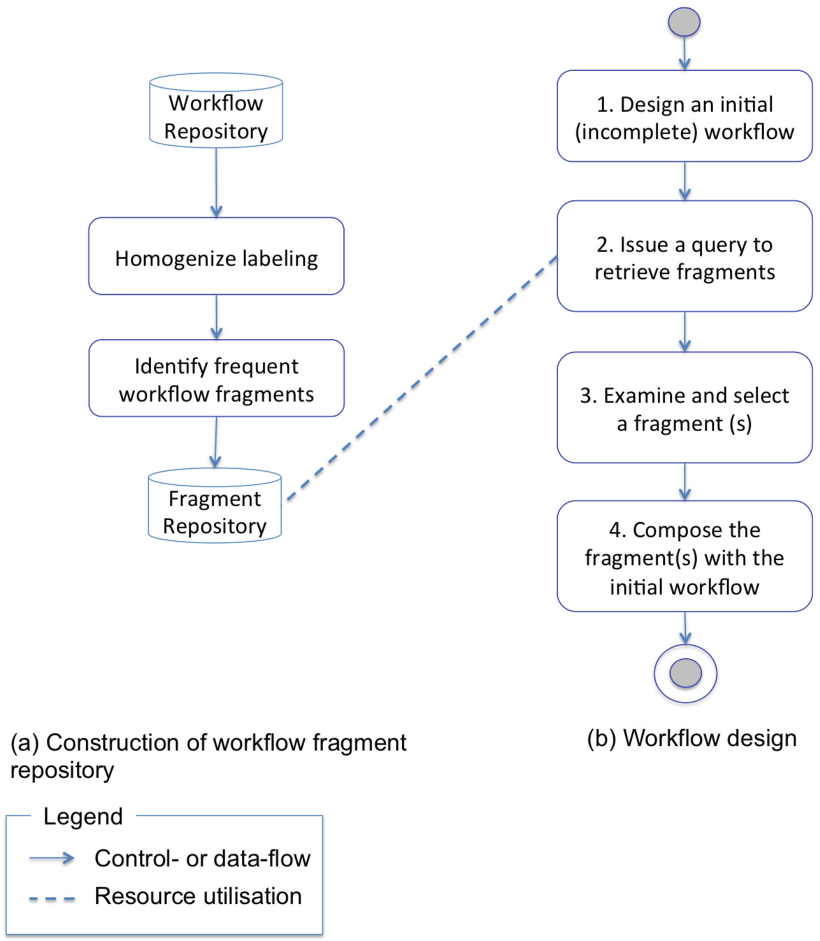

Workflow Repository
Homogenize labeling
Identify frequent workflow fragments
Fragment Repository
1. Design an initial (incomplete) workflow
2. Issue a query to retrieve fragments
3. Examine and select a fragment (s)
4. Compose the fragment(s) with the initial workflow
(a) Construction of workflow fragment repository
(b) Workflow design
Legend
Control- or data-flow
Resource utilisation

Fig. 1. An overview of the approach

we use sub-graph mining techniques, in particular the SUBDUE algorithm [11] to identify frequent fragments, which are stored in a dedicated repository. It is worth underlining that we only seek to mine frequent fragments since they are likely to represent patterns (and therefore best practices) that can be useful for the designer. Note also that our choice of Subdue is motivated by the popularity of this algorithm.

Figure 1(b) illustrates the process used to mine workflow fragments and exploit them when designing new workflows. In the first phase, the user starts by designing an initial workflow based on her objectives. There are some parts of the workflow that the designer may need assistance with. For a given part, which we name *fragment*, the designer issues a query against a repository of workflow fragments (phase 2). Such a query is composed of two elements: a set of keywords and a set of activities in the workflow being designed, which we name *joint activities*. The set of keywords are used to identify the fragments in the repository that are relevant. The joint activities specify the activities in the workflow being designed to which the relevant fragment(s) are to be connected to, to complete that workflow. This step returns a list of ranked candidates workflow fragments. The fragments are ranked based on the extent they match the keywords specified by the designer, but also based on their amenability to be connected to the joint activities in the workflow being designed. The user examines the top-k fragments and identifies the one or the ones she wishes to compose with the initial workflow (phase 3). Our system makes suggestions to the user on the way the composition can be performed (phase 4). The user accepts the suggestions she deems appropriate and complete the composition when necessary to obtain the desired workflow. We will present in more details the mining of frequent workflow fragments, their retrieval and their composition in Sects. 4, 5 and 6, respectively.

Workflow Model

For the work we present in this paper, we view a workflow as a graph $wf = (N, E)$, where N is a set of nodes composed of the activities A that constitute the workflow and control flow operators OP, i.e., $N = A \cup OP$. E represents the set of edges connecting activities and operators, i.e., $E \subseteq (N \times N)$. We consider the control flow operators supported by BPMN[1], namely sequence, and-split, and-join, or-split, or-join, xor-split and xor-split, the semantics of which is defined below.

- **Sequence:** The sequence flow connector (represented as an edge) is used to model the cases where the completion of the execution of an activity causes (or initializes) the execution of another activity.
- **And-split** and **And-join:** This operator is used when the completion of the execution of a given activity causes the execution of two or more activities, which are executed concurrently. The activities that are triggered by an and-split, or more precisely the workflow branches that are initialized by such activities, are usually synchronized by an and-join operator. Such an operator triggers the execution of the succeeding activity when the execution of given activities comes to completion.
- **Or-split** and **Or-join:** This operator is used when the completion of the execution of a given activity causes some or all of its subsequent activities within the workflow, which, as for and-split, are executed concurrently. The Or-join operator is usually used to synchronize the execution of workflow branches that were triggered by an or-split. It initializes the execution of the succeeding activity when the execution of some of the preceding activities terminates.

[1] http://www.bpmn.org.

– **Xor-split** and **Xor-join:** Unlike or-split, the xor-split is used when the activities (branches) that succeed a given activity are mutually exclusive. Therefore, during the execution one and only one of those activities are triggered. The xor-join is usually used to synchronize the branches that succeed an xor-split. It triggers the execution of the following activity once the execution of one of its preceding activities comes to completion.

Figure 2 illustrates a simple workflow composed of five activities. The workflow contains three control flow operators. The and-split connects the activity *att*1 to the activities *att*2 and *att*4, specifying that the execution of *att*2 and *att*4 is triggered once the execution of *att*1 terminates. The sequence connector is used to specify that the execution of *att*2 is followed by that of *att*3. Finally, the and-join operators connects the activities *att*3 and *att*4 to the activity *att*5, specifying that the execution of the latter is triggered once the execution of the formers terminates.

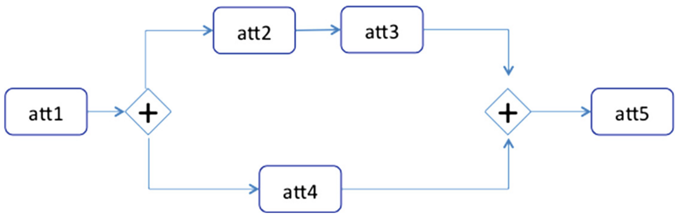


Fig. 2. A simple workflow

3 Related Work

There are three lines of work that are similar to our proposal which we analyze in this section and compare to our work: workflow similarity and mining, semantic enrichment as a means for improving workflow discovery and intelligent support for process modelling.

Workflow Similarity and Workflow Mining. The literature of business and scientific workflows is rich with proposals that seek to mine existing workflows and/or identify similarities between workflows. Existing work on workflow mining, focused mainly on deriving a workflow specification (usually as a petri-net) from logs of executions of the workflows. There are however some proposals that focused on examining workflow specifications using clustering [12] and case-based reasoning [13], among other techniques. For example, in the case of clustering-based techniques, several similarity measures were employed to estimate the distance between workflows. For example, Bae *et al.* [4] proposed a metric that uses tree structures to take into account control flow operators such as parallel branching and conditional choice. Diamantini *et al.* estimate the similarity between workflows based on the representation of workflows as Event Condition Action rules (ECA) [14]. Other authors apply sub-graph isomorphism techniques, see e.g. [5,6].

The above methods assess the similarity between entire workflows. In our work, we are interested in identifying the similarity between fragments of workflows. In this respect, our work is more related to the proposal by Diamantini *et al.* [14] who applied hierarchical graph clustering method in order to extract common fragments of workflows. We focus on fragments as opposed to entire workflows since there are more (realistic) opportunities for reuse at the level of the fragment as opposed to the entire workflow. In other words, the chances that the user finds a workflow that match her needs are slim. On the other hand, the chances that she finds workflows that contains one of more fragments that may contribute to her workflow are more likely.

Improving Similarity Using Semantic Enrichment. A comparative study of different methods for scientific workflow matching confirms that inclusion of external knowledge improves both computational complexity and result quality [15]. While the application of semantic enrichment has received notable attention in the literature as a means to enhance the quality of workflow matching, enhancing the quality of fragments matching has received less attention and is by and large unexplored.

One of the main issue that benefits from semantic enrichment is that of heterogeneity in naming the parameters of the workflows and their constituent activities. To do so, taxonomies are used to infer relationships between activities and their parameters [16,17]. We use a different strategy by augmenting existing sub-graph mining techniques [16–19] with a preprocessing phase for homogenizing workflow labels and by enriching user keywords query for searching fragments with synonyms. Trying to make a naive clustering of workflows in a repository would lead us to inefficient and limited method. Moreover, a striking distinction is that the previous works [17,18,20] propose the most reused fragments among the dataset as templates. Due to the large collection of available data on-line, the number of templates increases, which leads the designer to a heavy activity of browsing and analyzing the pool of templates in order to understand what could be useful to him. Instead, our work consists on assisting designer by using a simple keyword search to suggest the most probable component that could help her. We also offer support for integrating this fragment in her current workflow.

With the exception of the work by Diamantini *et al.* [14], we are not aware of any proposal that investigated the impact of the representation model used to encode workflows on the effectiveness and efficiency of workflow fragment mining. We proposed new models with respect to the work by Diamantini *et al.* We also conducted an empirical evaluation to investigate the advantages and limitations of each model. This study revealed that the representation model that we proposed out-preforms the remaining models both in terms of effectiveness and efficiency.

Advanced Support and Recommendation for Process Modelling. A categorization of recommendation techniques for process modelling is presented in [21], including textual recommendation, structural recommendation and linking recommendation. Following this classification, our technique would fall into the category of structural recommendation, that can be used for forward or backward completion. The approach in [22] aims also at helping designers to reuse parts of workflow models. In contrast with our approach, the user is required to describe the missing fragment using a dedicated query language.

Recommendation of an operator to extend a data analysis process is proposed in [23], based on a prediction model that is learned in a pre-processing phase using a pool of several thousand real-world data analysis workflows.

The notion of configurable operator that we use in our work is inspired by the works on configurable workflows (e.g., [24]), where it has a different goal, that of defining a generic process model whose behavior encompasses those of its variants. The configurable process model, resulting from the union of several alternative processes keeps information allowing analysts to track back, for each element, the process model form which it originates. Thus, the approach in [24] proposes to use it to construct the intersection of the process models, i.e., to identify common process fragments. A configurable business process is used also in [25] to create a Bayesian network to allow probabilistic recommendation queries.

Among the approaches for intelligent support for modelling, the work the most similar to the one presented in this paper is [26], which offers a search interface for process model fragments based on semantic annotations (tags). In contrast with our work, where process fragments are automatically extracted by mining the repository, users manually declare logically coherent process parts as fragments and assign titles to them.

To conclude, while similar works exist for different steps of our approach, the contribution presented in this paper is a complete and realistic solution for reusing process fragments starting from mining an heterogeneous repository using an efficient workflow encoding format, for offering a semantically enhanced keyword search, and for including support for their integration (composition) in a new workflow during the design task.

4 Mining Frequent Workflow Fragments

Given a repository of workflows, we would like to mine frequent workflow fragments, i.e., fragments that are used across multiple workflows. Such fragments are likely to implement tasks that can be reused in newly specified workflows. Mining workflows raises the following questions.

- *How to deal with the heterogeneity of the labels used by different users to model the activities of their workflows within the repository?* Different designers use different labels to name the activities that compose their workflow. We need a mean to homogenize the labels before mining the workflows in order not to miss relevant fragments.

– *Which graph representation is best suited for formatting workflows for mining frequent fragments?* We argue that the effectiveness and efficiency of the mining algorithm used to retrieve frequent fragments depend on the representation used to encode workflow specifications.

4.1 Homogenizing Activity Labels

To be able to extract frequent workflow fragments, we first need to identify common activities. Thus, activities implementing the same functionality should have the same names. Some workflow modelling tools (see for example Signavio[2]) handle a dictionary and allow to reuse dictionary entries via a search or by the auto-completion function (when user starts typing an activity label, the system suggests similar activity labels from the dictionary). If models in the repository come from different tools and use different naming conventions, a preprocessing step is applied to homogenize activity labels using a dictionary [27]. For facilitating this step, we rely on existing techniques like [28]. These techniques are able to recognize the labelling styles and to automatically refactor labels with quality issues. These labels are transformed into a verb-object label by the derivation of actions and business objects from activity labels. The labels of verb-object style contain an action that is followed by a business object, like *Create invoice* and *Validate order*. The benefits of this style of labeling have been promoted by several practical studies and modeling guidelines. Synonyms are also handled in this step. They are recognized using the WordNet lexical database and they are replaced with a common term.

4.2 Workflow Encoding

In order to extract frequent patterns, we use an existing graph mining algorithm, SUBDUE [11]. SUBDUE is a heuristic approach that uses a measure-theoretic information, the minimum description length, to find important subgraphs. Thus, the workflow model must be translated to a graph format. To this end, we studied some of the state of the art representation models and proposed one that can enhance the running time, the memory space required, and also the significance of the patterns extracted. In fact, we show that the representation model is of major importance.

The pre-processing phase consists of transforming a workflow into a compact graph representation. Indeed, the level of compactness depends on the representation model selected. As demonstrated by Diamantini *et al.* [14] and supported by our experimentation which we will report later on, the choice of the encoding model affects not only the time required for mining fragments, but also the relevance of the fragments returned.

In the experiment conducted by Diamantini and coauthors, three representation models A, B and C have been proposed (see Fig. 4). In all these models,

[2] www.signavio.com.

the activities are mapped to the so called activity nodes, while the representation of operators differs from one model to another. Specifically, in model A, each operator is represented by two nodes, called control nodes (to distinguish them from activities nodes): the first one is labeled "operator" and the second one is used to specify the type of operator. The labels that can be assigned to the latter one are: sequence, AND or XOR. The model A does not explicitly mention the difference between JOIN and SPLIT which can be deduced from the number of ingoing and outgoing arcs. The second model, model B assigns a control node to each operator, i.e., AND-split, XOR-split, AND-join, XOR-join. The operator SEQ is not explicitly represented by a node, instead it is translated into an edge connecting the activity nodes in question. Finally, the third model, C model simplifies the graph by removing both join and split nodes. XOR nodes are removed by generating a graph for each alternative path. In this way, the only nodes having several ingoing/outgoing activity are the AND nodes. As there is no ambiguity about these nodes, they can be removed also. Edges are labeled to maintain information about the type of the operator and its operand nodes.

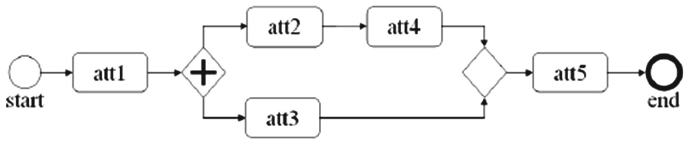


Fig. 3. An example of BPMN process (figure extracted from [14]).

According to the experiments conducted by Diamantini *et al.*, model A is the most costly in terms of execution time and also the less effective as it generates the least significant patterns. Indeed, when using representation model A the majority of nodes are control nodes. On the other hand, the model C contains no control node. The advantage of the model C is that the edges conveys information about the nodes attached to and the nature of the operator connecting them, which resulted in a gain in terms of storage space and execution time required. However, the disadvantage of model C lies in the mapping of the "XOR" operator; a graph is generated for each alternative which makes spatio-temporal complexity grow exponentially with the size of input data. Indeed, let us consider the case, where during the parsing of the original business process to convert into graph format the next node type is XOR-split with at least two incidents arcs. In this case, the number of graphs generated is doubled. In addition, if we consider an example repository where one of its most common substructures includes an operator XOR, this knowledge will not be discovered. Each path of the XOR operator will be extracted separately, but the fragment of business processes which contains all of these alternatives will not be considered as a whole. However, the model C is suitable for the discovery of typical pathways, which can be useful for some application domains.

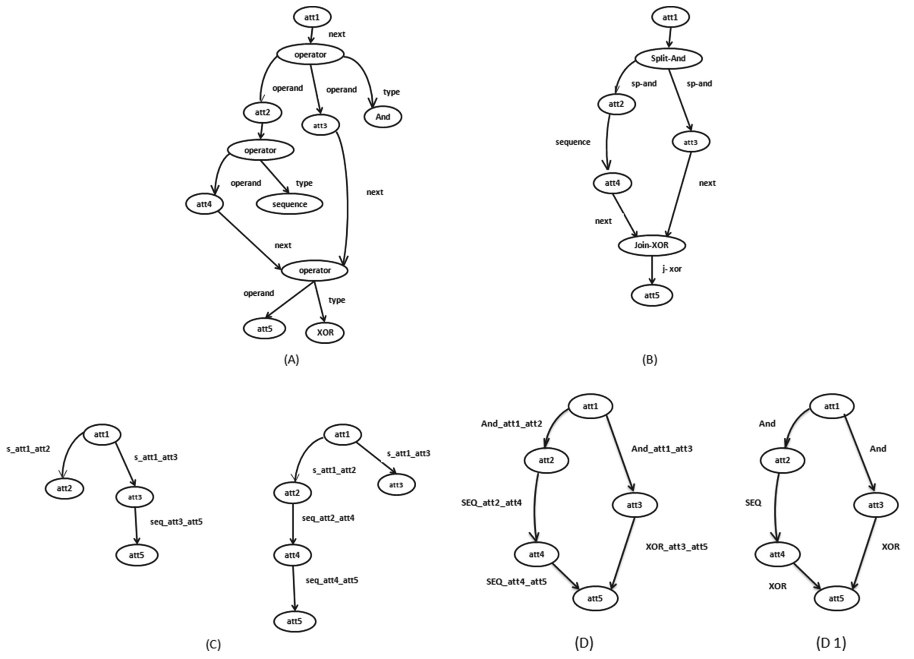


Fig. 4. Graphic of different representation models A, B, C and D.

Compared to model A and C, the model B has the higher level of compactness without a loss of information. In fact, representation model B reflects most closely the initial business process scheme. Therefore, patterns discovered based on this model are more interesting than those extracted by model A and C for searching and indexing cases.

We suggest a new representation model for workflows. We tried to take advantage of the previous models A, B and C and come up with a new model D that alleviates the disadvantages of such representation models. Specifically, we use the same strategy as the model C, in the sense that no control node is used. The edge connecting two activity nodes are labelled to indicate the kind of the control operator(s) connecting the nodes. We propose two variant representation models: D and D1. In the representation model D, the edges are labelled with the type of the operator and the labels of the activities that the operator connects. In the representation model D1, the edges are labeled only with the type of the control operator. That is, it does not consider the labels of the activities.

4.3 Empirical Evaluation

The methodology we have just described raises the following question: *Is the representation model that we propose suitable for mining frequent workflow fragments?* In this section we report on an empirical evaluation that we conducted to answer such a question.

Experimental Setting

We ran our experiment on a DELL machine with an Intel Core i7-2670QM processor with a 2.20 GHz frequency. We used the SUBDUE 5.5.2 tool in the fragment mining algorithm phase. We configured SUBDUE by choosing the MDL as a selection criterion with beam width equal to 4 and the number of top substructures returned set to 10.

We compared the representation models, namely A, B, C, D and D1. Our goal is to show, amongst other things, the drawbacks of the representation model C when it comes to dealing with datasets containing workflows with XOR operators. To do so, we generated three datasets composed of 30, 42 and 71 workflows. The datasets are composed of a mixture of some synthetic workflows that are obtained by mutating the workflow illustrated in Fig. 3 and some real ones selected from the Taverna 1 repository [1]. We manually examined the workflows generated to identify useful frequent fragments. Therefore, to measure the effectiveness of the representation models presented in this paper, we computed the precision and recall of the results obtained using each model.

Evaluation Results

Figure 5 illustrates the size of the graphs created using the different representation models. It shows that model A is the most expensive in term of space disk required to represent the dataset in graph format. Figure 6 compares the performances of the different representation models in terms of space disk, execution time and quality of results for different number of processes in the data set. Our results confirm those reported by Diamantnini *et al.* The A model requires the longest execution time (at least 7 times more than all other models); note this is not depicted in Fig. 6 for visibility reasons. Regarding the qualitative performance of the A model, we notice that when the number of workflows increases to 71, the recall decreases to 0%. This is due to the fact that control operators in the A model are represented by two nodes connected by an edge. For example the And-Split operator gives rise to two nodes connected by an edge: one is labelled "Operator" and the second "And-Split". Therefore, when mining the workflow repository using such a representation, the SUBDUE algorithm finds that the fragments representing control flow operators are the most frequent and returns them. However, they are useless for the designer as they do not implement any useful pattern that can be reused.

Concerning the C model, as expected, Fig. 5 shows that it may require more than twice the number of edges and nodes required by the models that we propose, namely D and D1. In addition, the model C is associated with a recall rate that varies between 32% and 34% for all tested databases which confirms that the C model can, at best, discover only one alternative at a time (in our case there are 2 alternatives attached to the XOR node).

Qualitatively, the B model performs much better than the A and C models. It retrieves successfully twice the number of significant elements retrieved using the C model and between 66% to 135% more than the A model. The B model was also able to discover more relevant workflow fragments than model D (about 10% more). This is due to the fact that the D model uses activity labels when labeling the edges. This over-specification of the labeling of the edges yields missing some fragments, and has therefore a negative impact on the recall. Note, however, that the B model returned more irrelevant workflow fragments (around 7%), which impacted negatively on the precision. Concerning the disk space requirements, the B model required between 25% up to 40% more space compared with the D and D1 models.

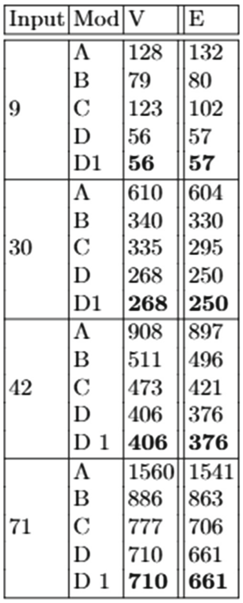
Based on the results illustrated in Fig. 6, we can observe a common precision performance between models D and D1. This performance is due to the fact that these two models do not use control nodes thereby avoiding retrieving false positive fragments, which will have a negative impact on the precision. We note, on the other hand, that the D1 model performs better than the D model when it comes to recall. This is due, as mentioned earlier, to the fact that the D model over-specifies the labels of the edges by using as well as the name of the control-operator, the labels of the activities connected by such an operator.

As SUBDUE loads the input data and performs all calculations in main memory, reducing the search space and the input file size, would also reduce the amount of memory required and computation time. Moreover, the D1 model also requires the smallest space compared with the other models. We can therefore conclude from this experiment that the D1 model was not only able to extract the most significant fragments but also did so in a relatively short execution time and required the least memory space. The performance achieved by the D1 model through this experiment has proven its effectiveness and efficiency.

5 Keyword-Based Search of Frequent Workflow Fragments

Given an initial workflow, the user issues a query to characterize the desired missing workflow fragment. The query is composed of two elements. The first element is a set of keywords $\{kw_1, \ldots, kw_n\}$ characterizing the functionality that the fragment should implement. The user selects the terms of her own choice. In other words, we do not impose any vocabulary for keyword specification. The second element in the user query specifies the activities in the initial workflow that are to be connected to the fragment in question, $A_{common} = \{a_1, \ldots, a_m\}$. Generally speaking, the user will specify one or two activities in the initial workflow. We call such activities using the terms common activities or joint activities, interchangeably.

The first step in processing the user query consists in identifying the fragments in the repository that are relevant given the specified keywords. In doing so, we adopt the widely used technique of TF/IDF (term frequency/inverse document frequency) measure. It is worth mentioning here that the workflow fragments in the repository are indexed by the labels of the activity obtained after the homogenization of the workflow (see Sect. 4.1). We refer to such an



Input	Mod	V	E
9	A	128	132
	B	79	80
	C	123	102
	D	56	57
	D1	**56**	**57**
30	A	610	604
	B	340	330
	C	335	295
	D	268	250
	D1	**268**	**250**
42	A	908	897
	B	511	496
	C	473	421
	D	406	376
	D 1	**406**	**376**
71	A	1560	1541
	B	886	863
	C	777	706
	D	710	661
	D 1	**710**	**661**

Fig. 5. Size of the graphs using the different representation models.

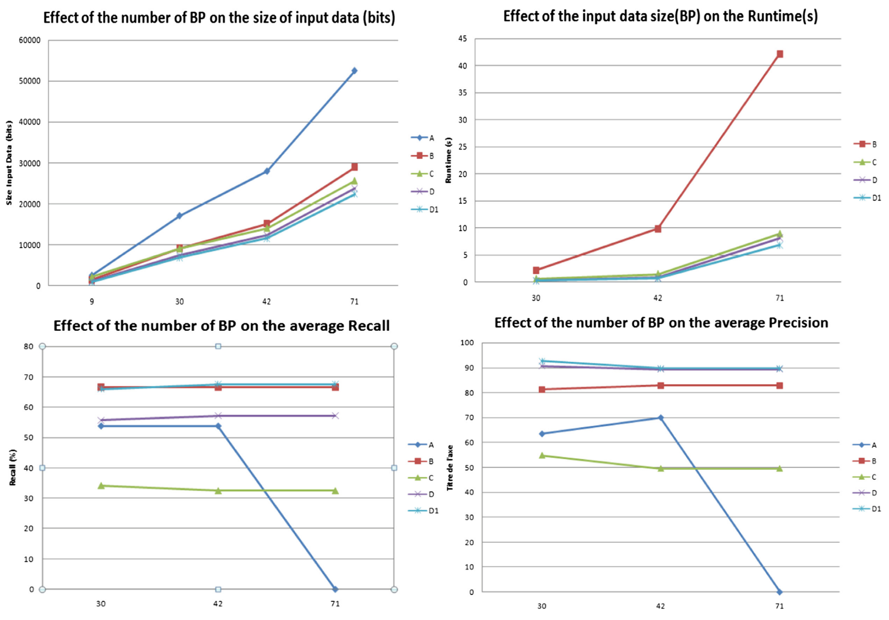


Fig. 6. Performances of the different representation models.

index by IDX. Applying directly TF/IDF based on the set of keywords provided by the user is likely to miss some fragments. This is because the designer provides keywords of her own choosing; a given keyword may not be present in the repository, while one of its synonyms could be present. To address this issue, we adopt the following method which augments traditional TF/IDF with a semantic enrichment phase. Specifically, given a set of keywords provided by the user $\{kw_1, \ldots, kw_n\}$, for each keyword kw_i we retrieve its synonyms, which we denote by $syn(kw_i)$ from an existing thesaurus, e.g., Wordnet [29] or a specialized domain specific thesaurus if we are dealing with workflows from a specific domain. The index IDX is trawled to identify if there is a term in $syn(kw_i) \cup \{kw_i\}$ that appears. If this is the case, then the term in the index is used to represent kw_i in the vector that will be used to compare the query with the vectors representing the workflow fragments in the repository.

Note also that multiple keywords, e.g., kw_i and kw_j, may be mapped to the same term kw_{IDX} in the index IDX, in this case we set the frequency of the kw_{IDX} to be the number of keywords it represents when computing the associated TF/IDF. In certain situations, we may need to use the hypernyms of kw_i. This is specifically the case when none of the terms in IDX appears in $syn(kw_i) \cup \{kw_i\}$. Using the hypernyms in such a case may allow us to identify a term in IDX that can be used to represent kw_i in the vector representing the user query.

Once the vector that represents the user query is constructed and the TF/IDF of its associated terms are calculated, it is compared against the vectors representing the workflow fragments in the repository using the cosine similarity [30].

The set of fragments retrieved in the previous step, which we call candidate fragments, are then examined. Specifically, their constituent activities are compared and matched against the activities in A_{common}, that are specified by the user. Given a candidate fragment wf_{frg}, each activity a_j in A_{common} is compared (matched) against the activities that constitute the fragment wf_{frg}. The objective of this step is to identify the activities that will be in common between the fragment and the initial workflow. Note that for a given activity a_j in A_{common} there may be more than one matching activity in the fragment wf_{frg}. In this case, we associate a_j with the activity in wf_{frg} with the highest matching score. Reciprocally, if two activities a_i, a_j in A_{common} are associated with the same activity in wf_{frg}, the one with the highest score is kept and other matcher is searched for the second one (such that the sum of the similarities of the global mapping is maximised). Note also that it is possible that a_j may not have any matching activities among those of the fragment wf_{frg}. The matching is performed using existing techniques for matching activity labels [31]. These techniques tokenize the activity labels, remove stop words and then apply syntactic string comparisons (string edit distance, number of common words) and take semantic relationships into account based on the lexical database WordNet.

The last step in the query processing consists in ranking the candidate fragments. The ranking takes into consideration the following factors.

1. The relevance score of the candidate fragment calculated based on TF/IDF given a user query uq. (We view a fragment as a document, or more specifically, we consider the terms labeling the activities in the fragment when computing the TF/IDF.) We use $Relevance(wf_{frg}, uq)$ to denote the relevance score associated with a candidate fragment wf_{frg} given the user query.
2. The frequency of use of the fragment in the repository. The fragment that are used across multiple workflows are likely to implement best practices that are useful for the workflow designer compared with workflow fragment that are used in, say, only 2 workflows. We use $Frequency(wf_{frg})$ to denote the frequency, i.e., the number of times a candidate fragment wf_{frg} appears in the mined workflow repository.
3. The compatibility of the candidate fragment with the inital workflow. To estimate the compatibility, we consider the number of activities in A_{common} that have a matching activity in the workflow fragment and their associated matching score. Specifically, we define the compatibility of a workflow fragment given the activities $uq.A_{common}$ specified in the user query uq as follows:

$$Compatibility(wf_{frg}, uq.A_{common}) = \frac{\sum_{a_j \in A_{common}} matchingScore(a_j, wf_{frg})}{|A_{common}|}$$

   where $matchingScore(a_j, wf_{frg})$ is the matching score between a_j and the best matching activity in wf_{frg}, and $|A_{common}|$ the number of activities in A_{common}. $Compatibility(wf_{frg}, A_{common})$ takes a value between 0 and 1. The larger is the number of activities A_{common} that have a matching activity in wf_{frg} and the higher are the matching scores, the higher is the compatibility between the candidate workflow fragment and the initial workflow.

Based on the above factors, we define the score used for ranking candidate fragments given a user query uq as:

$$Score(wf_{frg}, uq) =$$

$$w_r.Relevance(wf_{frg}, uq) + w_f.\frac{Frequency(wf_{frg})}{MaxFrequency} + w_c.Compatibility(wf_{frg}, uq.A_{common})$$

where w_r, w_f and w_c are positive real numbers representing weights such that $w_r + w_f + w_c = 1$. $MaxFrequency$ is a positive number denoting the frequency of the fragment that appears the maximum number of times in the workflow repository harvested. Notice that the score takes a value between 0 and 1. Once the candidate fragments are scored, they are ranked in the descendant order of their associated scores. The top-k fragments, e.g., 5, are then presented to the designer who selects to the one to be composed with the initial workflow. Initially the weights w_r, w_f and w_c take equal values. Then, depending on the performance and the feedback provided by the user they can be adjusted.

6 Composing Workflow Fragments

Once the user has examined the fragments that are retrieved given the set of keywords she specified, she can choose a fragment to be composed with the initial workflow she was designing. We present in this section a method that can assist the designer in the composition task. Specifically, we consider that the user has designed an initial workflow $wf_{initial}$ and selected a fragment wf_{frg} to be composed with $wf_{initial}$. We turn our attention first to the case where $wf_{initial}$ and wf_{frg} has one activity in common a_{common}. We denote by $in(a_{common}, wf)$ the set of activities that precedes a_{common} in the workflow wf, and by $out(a_{common}, wf)$ the set of activities that succeed a_{common} in the workflow wf.

```
Algorithm Compose
Input:  $wf_{initial}$ initial workflow
        $wf_{frg}$: workflow fragment
        $a_{common}$: activity in common between $wf_{initial}$ and $wf_{frg}$
Output: $wf_{merge}$: workflow obtained by composing $wf_{initial}$ and $wf_{frg}$ based on $a_{common}$
Begin
1       If $(card(out(a_{common}, wf_{initial})) = 0)$ and $(card(in(a_{common}, wf_{frg})) = 0)$
2           Then connect in sequence $wf_{initial}$ followed by $wf_{frg}$ using $a_{common}$
3       If $(card(in(a_{common}, wf_{initial})) = 0) and (card(out(a_{common}, wf_{frg})) = 0)$
4           Then connect in sequence $wf_{frg}$ followed by $wf_{initial}$ using $a_{common}$
5       If $(card(in(a_{common}, wf_{initial})) \geq 1) and (card(in(a_{common}, wf_{frg})) \geq 1)$
6           Then create a configurable join operator and
7                connect it to the preceeding activities of $wf_{initial}$ and those of $wf_{frg}$
8       If $(card(out(a_{common}, wf_{initial})) \geq 1) and (card(out(a_{common}, wf_{frg})) \geq 1)$
9           Then create a configurable branching operator and
10               connect it to the succeeding activities of $wf_{initial}$ and those of $wf_{frg}$
11      $wf_{merge}$ is the workflow obtained as a result of the above manipulation.
End
```

Fig. 7. Composition algorithm

Figure 7 sketches the algorithm used for composing $wf_{initial}$ and wf_{frg}. If there is no succeeding activity for a_{common} in wf_{intial} and there is no preceding activity for a_{common} in wf_{frg} then $wf_{initial}$ is composed in sequence with wf_{frg} based on a_{common} (*lines 1 and 2*). Inversely, if there is no preceding activity for a_{common} in $wf_{initial}$ and there is no succeeding activity for a_{common} in wf_{frg} then wf_{frg} is composed in sequence with $wf_{initial}$ based on a_{common} (*lines 3 and 4*). The above cases are illustrated using Fig. 8.

If a_{common} has preceding activities in both $wf_{initial}$ and wf_{frg} (*line 5*), then we connect the two workflows using a configurable join operator as illustrated in Fig. 8 (*lines 6 and 7*). We call such an operator configurable because it is up to the user to choose if such an operator is of a type $and - join$, $or - join$ or $xor - join$. The preceding activities of such an operator are the preceding activities of $wf_{initial}$ and wf_{frg}, and its succeeding activity is a_{common} (Fig. 9).

If a_{common} has succeeding activities in both $wf_{initial}$ and wf_{frg} (*line 8*), then we use a configurable split operator to connect the two workflows as illustrated in Fig. 10 (*lines 9 and 10*). We call such an operator configurable because it is up to the user to choose if such an operator is of a type *and-split*, *or-split* or *xor-split*. The preceding activity of such an operator is a_{common} and its succeeding activities are the succeeding activities of $wf_{initial}$ and wf_{frg}.

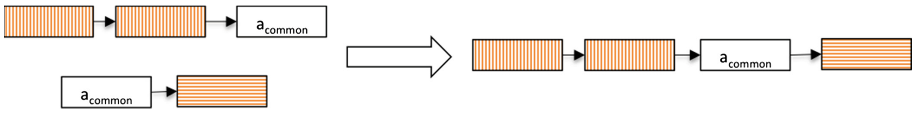


Fig. 8. Merging the initial workflow and a fragment in sequence based on the common activity a_{common}.

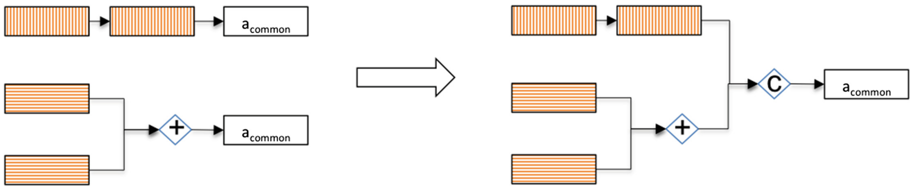


Fig. 9. Merging the preceding activities of the initial workflow and a fragment using a configurable join control operator.

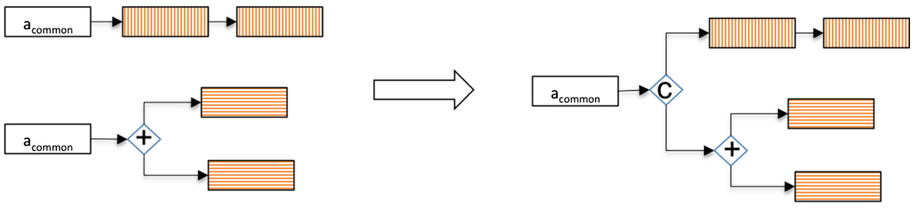


Fig. 10. Merging the succeeding activities of the initial workflow and a fragment using a configurable split control operator.

If the initial workflow and the fragment has more than one activity in common then we perform the processing we have just described above iteratively using one activity at a time. Note, however, that one would expect in the general case that there is one activity in common between the initial workflow and the fragment selected by the user. Having multiple activities in common between the initial workflow and the fragment is likely to lead to complex workflows that are difficult to understand thereby out-weighting the benefits that can be derived from fragment reuse. Once the initial workflow and the fragment selected are merged, the user examines the obtained workflow, makes changes in terms of activities and control flow connectors if necessary. In particular, the user will need to substitute configurable join and split operators with concrete operators, e.g., and-split or or-split, that meet the semantics of the process she has in mind. As mentioned earlier, the user may want to merge another fragment once the initial workflow and the current fragment has been merged. The same processing described above will be applied to the newly selected fragment.

7 Example from eScience

In this section, we illustrate the use of the method we have proposed in this paper to assist a designer of a workflow from the eScience field. Specifically, we show how the method described can help the designer specify a workflow that is used for analyzing Huntington's disease (HD) data. Huntington's disease is the most common inherited neurodegenerative disorder in Europe, affecting 1 out of 10000 people. Although the genetic mutation that causes HD was identified 20 years ago [32], the downstream molecular mechanisms leading to the HD phenotype are still poorly understood. Relating alterations in gene expression to epigenetic information might shed light on the disease aetiology. With this in mind, the scientist wanted to specify a workflow that annotate HD gene expression data with publicly available epigenetic data [33].

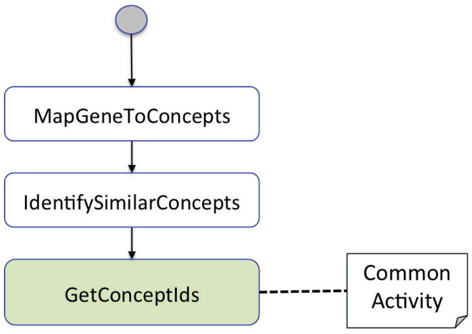


Fig. 11. Initial Huntington disease profiling workflow.

The initial workflow specified by the designer is illustrated in Fig. 11. The workflow contains three activities that are ordered in sequence. The first activity *MapGeneToConcept* is used to extract the concepts that are used in domain ontologies to characterize a given gene that is known or suspected to be involved in an infection or disease, in this case the Huntigton Disease. The second activity *IndentifySimilarConcepts* is then used to identify for each of those concepts, the concepts that are similar. The rational behind doing so is that genes that are associated with concepts that are similar to concepts that are involved in the disease have chances of being also involved in the disease. This activity involves running ontological similarity tests to identify similar concepts. The third activity *GetConceptIds* is then used to extract the identifiers of the similar concepts.

The workflow depicted in Fig. 11 does not completely implement the analysis that the designer would like. In particular, the designer would like to profile and rank the similar concepts that have been identified by the third activity but is unaware of the service implementations that can be used to do so. To assist the designer in this task, we asked him to identify the common activity to which the missing fragment is to be attached to. She identified the activity *GetConceptIDs* as the common activity. We then asked him to provide a set of keywords characterizing the desired fragment. She provided as a result the following set: $\{concept, score, rank, profile\}$. Using the method presented in Sect. 5, we retrieved candidate fragments. Figure 12 illustrates the fragment that was first ranked in the results and was selected by the user. The common activity is named $GetTermIdentifiers$ in the fragment, which is different from the label of the common activity in the initial workflow, *GetConceptIds*. Still, our method was able detect it as a common activity thanks to the string similarity utilized.

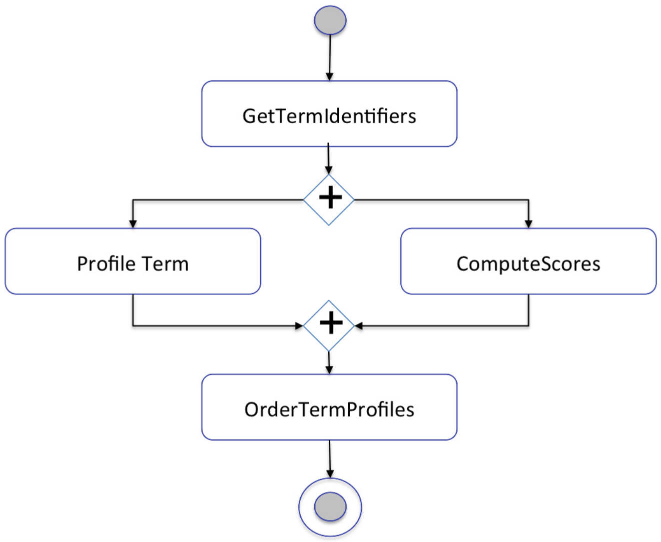


Fig. 12. Candidate fragment selected by the workflow designer.

By applying the composition algorithm presented in Sect. 6 to merge the initial workflow and the fragment selected by the user, we obtained the workflow depicted in Fig. 13. Indeed, the common activity has no succeeding activity in the initial workflow and has no preceding activity in the workflow fragment (*lines 1, 2* in Fig. 7). Therefore, the two workflows are connected ins sequence using the common activity.

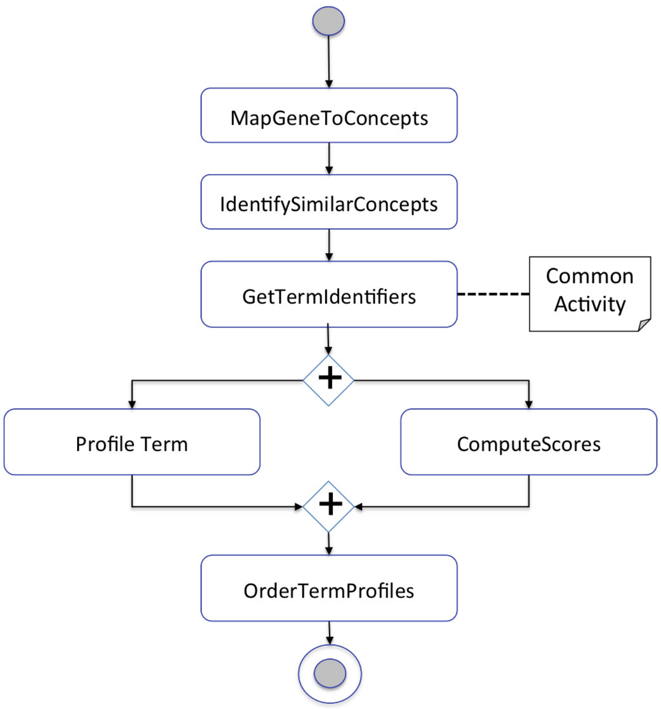


Fig. 13. Workflow obtained by merging the initial workflow and selected fragment.

8 Conclusion

We presented in this paper a methodology for improving the reusability of fragments within workflow repositories, with the objective of allowing workflow designers to benefit from existing workflows (and the knowledge they encompass) when designing new workflows. Specifically, we examined the representation model that can be used for formatting workflows before they are mined. In order to propose a realistic and complete solution, we showed also how to deal with the heterogeneity of activity labels as a preprocessing step before mining. The experimentation shows the effectiveness of the representation model in improving the performance of the mining task. The mined fragments can be searched by designers using a simple free keyword search and automatically integrated in the initial workflow model. Our ongoing work aims to examine how several fragments from different workflow specifications can be combined to meet user needs. We also intend to perform a larger scale evaluation to assess the performance of the solution proposed and a user study to evaluate the efficiency gain of a designer using our approach.

References

1. De Roure, D., Goble, C.A., Stevens, R.: The design and realisation of the my$_{\text{experiment}}$ virtual research environment for social sharing of workflows. Future Gener. Comput. Syst. **25**(5), 561–567 (2009)
2. Mates, P., Santos, E., Freire, J., Silva, C.T.: CrowdLabs: social analysis and visualization for the sciences. In: Cushing, J.B., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 555–564. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22351-8_38
3. Giardine, B., Riemer, C., Hardison, R.C., Burhans, R., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Miller, W., Kent, W.J., Nekrutenko, A.: Galaxy: a platform for interactive large-scale genome analysis. Genome Res. **15**, 1451–1455 (2005)
4. Bae, J., Caverlee, J., Liu, L., Yan, H.: Process mining by measuring process block similarity. In: Eder, J., Dustdar, S. (eds.) BPM 2006. LNCS, vol. 4103, pp. 141–152. Springer, Heidelberg (2006). doi:10.1007/11837862_15
5. Goderis, A., Li, P., Goble, C.: Workflow discovery: the problem, a case study from e-science and a graph-based solution. In: International Conference on Web Services, ICWS 2006, Chicago, IL, pp. 312–319. IEEE (2006)
6. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. Inf. Syst. **40**, 115–127 (2014)
7. Harmassi, M., Grigori, D., Belhajjame, K.: Mining workflow repositories for improving fragments reuse. In: Cardoso, J., Guerra, F., Houben, G.-J., Pinto, A.M., Velegrakis, Y. (eds.) KEYSTONE 2015. LNCS, vol. 9398, pp. 76–87. Springer, Cham (2015). doi:10.1007/978-3-319-27932-9_7
8. Deutch, D., Milo, T.: Evaluating TOP-K queries over business processes. In: Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, Shanghai, China, 29 March 2009–2 April 2009, pp. 1195–1198 (2009). http://dx.doi.org/10.1109/ICDE.2009.199
9. Goderis, A., Li, P., Goble, C.A.: Workflow discovery: requirements from e-science and a graph-based solution. Int. J. Web Serv. Res. **5**(4), 32–58 (2008). http://dx.doi.org/10.4018/jwsr.2008100102
10. Starlinger, J., Brancotte, B., Boulakia, S.C., Leser, U.: Similarity search for scientific workflows. PVLDB **7**(12), 1143–1154 (2014). http://www.vldb.org/pvldb/vol7/p.1143-starlinger.pdf
11. Jonyer, I., Cook, D.J., Holder, L.B.: Graph-based hierarchical conceptual clustering. J. Mach. Learn. Res. **2**, 19–43 (2001)
12. Yaman, M.B.F., Oates, T.: A context driven approach for workflow mining. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, pp. 1798–1803. Morgan Kaufmann Publishers Inc. (2009)
13. Leake, D., Kendall-Morwick, J.: Towards case-based support for e-science workflow generation by mining provenance. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 269–283. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85502-6_18
14. Diamantini, C., Potena, D., Storti, E.: Mining usage patterns from a repository of scientific workflows. In: Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, 26–30 March 2012, pp. 152–157. ACM (2012). http://doi.acm.org/10.1145/2245276.2245307

15. Starlinger, J., Brancotte, B., Cohen-Boulakia, S., Leser, U.: Similarity search for scientific workflows. In: 40th International Conference on Very Large Data Bases, VLDB Endowment, Hangzhou, China, pp. 2150–8097 (2014)
16. Cuzzocrea, A., Diamantini, C., Genga, L., Potena, D., Storti, E.: A composite methodology for supporting collaboration pattern discovery via semantic enrichment and multidimensional analysis. In: 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR), Tunis, Tunisa, pp. 459–464. IEEE (2014)
17. Garijo, D., Corcho, Ó., Gil, Y.: Detecting common scientific workflow fragments using templates and execution provenance. In: Proceedings of the Seventh International Conference on Knowledge Capture, pp. 33–40. ACM, New York (2013)
18. Diamantini, C., Genga, L., Potena, D., Storti, E.: Innovation pattern analysis. In: 2013 International Conference on Collaboration Technologies and Systems (CTS), San Diego, CA, pp. 628–629. IEEE (2013)
19. Garijo, D., Corcho, Ó., Gil, Y., Gutman, B.A., Dinov, I.D., Thompson, P.M., Toga, A.W.: Fragflow automated fragment detection in scientific workflows. In: 10th IEEE International Conference on e-Science, Sao Paulo, Brazil, pp. 281–289. IEEE (2014)
20. Diamantini, C., Genga, L., Potena, D., Storti, E.: Discovering behavioural patterns in knowledge-intensive collaborative processes. In: Appice, A., Ceci, M., Loglisci, C., Manco, G., Masciari, E., Ras, Z.W. (eds.) NFMCP 2014. LNCS, vol. 8983, pp. 149–163. Springer, Cham (2015). doi:10.1007/978-3-319-17876-9_10
21. Kluza, K., Baran, M., Bobek, S., Nalepa, G.J.: Overview of recommendation techniques in business process modeling. In: Proceedings of 9th Workshop on Knowledge Engineering and Software Engineering (KESE9) Co-located with the 36th German Conference on Artificial Intelligence (KI 2013), Koblenz, Germany, 17 September 2013
22. Awad, A., Sakr, S., Kunze, M., Weske, M.: Design by selection: a reuse-based approach for business process modeling. In: Jeusfeld, M., Delcambre, L., Ling, T.-W. (eds.) ER 2011. LNCS, vol. 6998, pp. 332–345. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24606-7_25
23. Jannach, D., Jugovac, M., Lerche, L.: Adaptive recommendation-based modeling support for data analysis workflows, pp. 252–262 (2015)
24. Rosa, M.L., Dumas, M., Uba, R., Dijkman, R.M.: Business process model merging: an approach to business process consolidation. ACM Trans. Softw. Eng. Methodol. **22**(2), 11 (2013)
25. Bobek, S., Nalepa, G.J., Grodzki, O.: Integration of activity modeller with Bayesian network based recommender for business processes. In: Proceedings of 10th Workshop on Knowledge Engineering and Software Engineering (KESE10) Co-located with 21st European Conference on Artificial Intelligence (ECAI 2014), Prague, Czech Republic, 19 August 2014
26. Koschmider, A., Hornung, T., Oberweis, A.: Recommendation-based editor for business process modeling. Data Knowl. Eng. **70**(6), 483–503 (2011)
27. Peters, N., Weidlich, M.: Automatic generation of glossaries for process modelling support. Enterp. Model. Inf. Syst. Archit. **6**(1), 30–46 (2011)
28. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Inf. Syst. **37**(5), 443–459 (2012)
29. Princeton University (2010) About WordNet. http://wordnet.princeton.edu/wordnet/

30. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern information retrieval - the concepts and technology behind search, 2nd edn. Pearson Education Ltd., Harlow (2011). http://www.mir2ed.org/
31. Cayoglu, U., et al.: Report: the process model matching contest 2013. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 442–463. Springer, Cham (2014). doi:10.1007/978-3-319-06257-0_35
32. The Huntington's Disease Collaborative Research Group: A novel gene containing a trinucleotide repeat that is expanded and unstable on Huntington's disease chromosomes. Cell **72**(6), 971–983 (1993)
33. Mina, E., van Roon-Mom, W., 't Hoen, P.A., Thompson, M., van Schouwen, R., Kaliyaperumal, R., Hettne, K., Schultes, E., Mons, B., Roos, M.: Prioritizing hypotheses for epigenetic mechanisms in Huntington's disease using an e-science approach. J. BioData Min. (2014, submitted)