FragFlow: Automated Fragment Detection in Scientific Workflows

Daniel Garijo, Oscar Corcho
Ontology Engineering Group
Dpto. Inteligencia Artificial,
Facultad de Informática
Universidad Politécnica
de Madrid
{dgarijo, ocorcho}@fi.upm.es

Yolanda Gil
Information Sciences Institute
and Department of Computer
Science
University of Southern
California
gil@isi.edu

Boris A. Gutman, Ivo D. Dinov,
Paul Thompson and
Arthur W. Toga
Laboratory of Neuro Imaging
Institute for Neuroimaging and
Informatics
Keck School of Medicine
University of Southern California

***Abstract*—Scientific workflows provide the means to define, execute and reproduce computational experiments. However, reusing existing workflows still poses challenges for workflow designers. Workflows are often too large and too specific to reuse in their entirety, so reuse is more likely to happen for fragments of workflows. These fragments may be identified manually by users as sub-workflows, or detected automatically. In this paper we present the FragFlow approach, which detects workflow fragments automatically by analyzing existing workflow corpora with graph mining algorithms. FragFlow detects the most common workflow fragments, links them to the original workflows and visualizes them. We evaluate our approach by comparing FragFlow results against user-defined sub-workflows from three different corpora of the LONI Pipeline system. Based on this evaluation, we discuss how automated workflow fragment detection could facilitate workflow reuse.**

I. Introduction

Scientific workflows are templates that define the set of steps and data dependencies needed to execute computational experiments. They are usually created for executing research methods and sharing them with a community of users [21] [8], even in cases where the community is small (e.g., a research lab). Scientific workflows can ease data management among different steps [9] [15], facilitate collaborative development through a common interface and simplify access to computational resources (e.g., clusters, grids, etc.).

Different scientific workflows often share part of their functionality (common preprocessing steps, data manipulation for a particular visualization, reformatting, etc). A prior study analyzing workflows from different workflow systems and domains showed that almost 20% percent of the analyzed workflows were composed of other workflows [6]. Some systems even allow users to define hierarchical workflows to exploit these commonalities and split workflows into smaller reusable parts [15] [21]. However, there is not much support for the automatic detection of sub-workflows at the moment.

In our work, we aim to automatically discover and expose common workflow fragments from a given corpus of workflows. Capturing common workflow fragments may bring in several benefits to workflows users and designers: they simplify the visualization of the workflow (simpler visualizations lead to better organization and comprehension), they make the workflow easier to understand, they modularize the functionality (by separating the different reusable parts of the workflow) and they save time in workflow design (designers are more likely to reuse fragments rather than complete workflows, which are more specific).

Our approach combines exact and inexact graph mining techniques to find the most common workflow fragments in a corpus of workflows. Our implementation, FragFlow[1], builds on previous work [5], integrating additional graph mining algorithms and expanding it with filters, data preparation and statistical analysis steps. FragFlow can be configured to find fragments of different sizes or frequencies, visualize them and link them back to the original corpus of workflows.

We evaluate our approach by comparing automatically detected workflow fragments to sub-workflows (groupings) created by users of the LONI Pipeline, a workflow system for neuroimaging analysis [21].

This paper is structured as follows. Section II introduces the main goals for this work. Section III describes in detail the workflow mining algorithms for detecting fragments, while Section IV explains how they are used in FragFlow. Section V introduces details on the experimental setup (corpora used and evaluation metrics); and Section VI provides the results of the evaluation. Finally, Section VII introduces related work and we finish by discussing conclusions and future work in Section VIII.

II. Goals

This section defines the main objectives of our work. First we introduce the terminology used in the paper, and then we describe our main goals.

We define a *workflow fragment* as a set of connected steps that are part of a workflow. We define *common workflow fragments* as those fragments that occur more than once in a corpus of workflows. The higher the frequency they occur, the

[1] http://dx.doi.org/10.5281/zenodo.11219

more useful the workflow fragment may be, as they may have more potential for reuse. If a workflow is reused as a whole in another workflow, it becomes a workflow fragment. We refer to *FragFlow workflow fragments* as those common workflow fragments detected automatically by our FragFlow system.

We define a *grouping* as a workflow fragment manually identified by a user. A grouping is likely to be included in other workflows, but it is not always the case. A grouping may also have nested *sub-groupings*, which may or may not be included in another workflow or grouping. Groupings are likely to have an explicit function associated to them, explaining their main role in the workflow.

An example of a workflow with groupings is depicted in Figure 1, where the main steps of the Minimal Deformation Target workflow (a workflow for creating an "average" image from a group of 3D images of the brain) are shown. The workflow was designed in the LONI Pipeline, a workflow system designed to create workflows in neuroimaging research. Figure 1 shows three groupings with different functionality (align linear, reconcile and define common; KL_MI3-step multiscale and Composite Geometric Averaging). Each of these groupings contains several steps. In the figure, the top grouping has been expanded to show its inner steps. In this workflow, each of the groupings occurs once. As the LONI Pipeline allows users to create groupings, we define *LONI Pipeline groupings* as those groupings created by users in the LONI Pipeline environment.

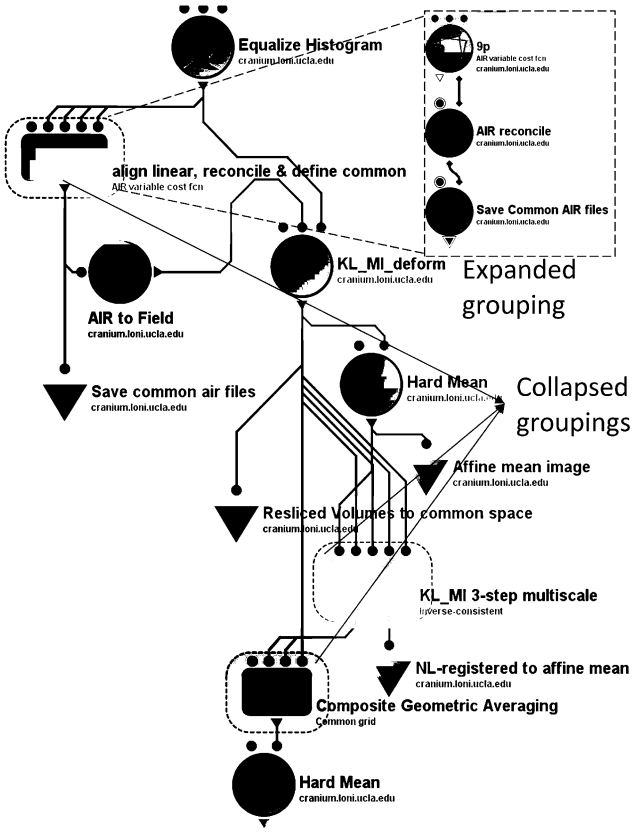

Figure 1: Snapshot of the Minimal Deformation Target workflow (MDT). Three groupings have been highlighted (collapsed), and one of them (on top) has been expanded to show three inner components. The function of the groupings is exposed in their labels.

The goal of this work is to answer the following questions:

a) Are automatically detected workflow fragments similar to user-defined groupings? Users are likely to group parts of workflows that they reuse in other workflows. By comparing automatically detected FragFlow workflow fragments against user-defined LONI Pipeline groupings, we will be able to determine whether FragFlow fragments may be suggested automatically to the user as they create new workflows.

b) For those automatically detected workflow fragments that are not similar to user-defined groupings, do users find them useful? Users create groupings in different manners. Our approach may find additional or alternative groupings to those originally defined manually by users.

c) How are workflows and groupings reused? In this work we assume that workflows will be reused, based on a previous study in different workflow systems [6]. By measuring the size, distribution and number of LONI Pipeline groupings, we can assess how groupings and templates are reused, and which design principles are followed by workflow designers (e.g., small groupings for simplicity, big groupings that can be pasted into other workflows, etc.).

III. Workflow Fragment Detection Algorithms

Our approach is based on the application of graph mining techniques to a workflow corpus. This section introduces the algorithms that have been integrated in our system, FragFlow, which expands our previous work [5] and also includes result filtering, visualization and statistics of the common obtained fragments.

We represent workflows as labeled directed acyclic graphs (LDAGs). We reuse mining techniques for Frequent Graph Matching detection (FGM). At the moment FragFlow integrates three different algorithms that use two types of graph mining techniques:

Inexact FGM Techniques: techniques that use approximate measures to calculate the similarity between two graphs [12], and detect the commonalities among them. In general, these techniques apply heuristics to detect the most common workflow fragments efficiently. However, the solution is not always complete. Therefore, these techniques do not identify all the possible common fragments in the corpus. FragFlow integrates the SUBDUE algorithm [3], which applies graph clustering on the input corpus, and provides two different heuristics for detecting common fragments:

- *Minimum description length (MDL):* at each iteration, the best fragment is chosen according to the bytes necessary to encode the input graph collection.
- *Size*: at each iteration, the best fragment is chosen according to how the overall size of graph collection is reduced.

Exact FGM Techniques: techniques that deliver all the possible frequent sub-graphs included in a dataset. FragFlow integrates two different algorithms:

- *The gSpan algorithm* [22], one of the most popular exact FGM techniques, by using a deep first search strategy and canonical representation for each graph.
- *FSG* [13], which uses a breadth first strategy and a canonical labeling method for graph comparison.

Since FSG works only on undirected labeled graphs, some of the fragments it finds on our LDAGs are incorrect. We are extending FragFlow to filter FSG results, but we have not included this algorithm in the evaluation of this work.

A. Exact FGM versus inexact FGM techniques

Both exact and inexact FGM techniques have advantages and disadvantages. On the one hand, inexact FGM techniques tend to find smaller but highly frequent patterns, although they might not identify all potential useful workflow fragments since they are incomplete.

On the other hand, exact FGM techniques identify all possible fragments in the input corpus. However, when the frequent sub-graph size in the input dataset is high, exact FGM techniques might return too many fragment candidates, while consuming significant computational resources. Adjusting the parameters of the search (e.g., frequency of the desired workflow fragments) is important in order to obtain the best results.

B. Frequency-based versus transaction-based techniques

Another important aspect is how each technique considers the graphs of the input corpus. Some approaches are *frequency-based*, i.e., they consider each candidate structure based on the number of times they appear (frequency), while others are *transaction-based*, i.e., they only consider a candidate structure once per input graph (support). For example, if a three step sequence appears 200 times in a workflow, a frequency based approach would consider the sequence of steps as a candidate fragment that appeared 200 times, while a transaction based approach would count it as one. Both techniques produce valuable workflow fragments, and are worth considering for detecting workflow common fragments.

In FragFlow all the currently integrated inexact FGM techniques are frequency based, while exact FGM techniques are transaction based. In this work we compare inexact and exact FGM to determine the best one to apply on each corpus.

IV. Fragment Detection in FragFlow

Figure 2 shows the major components of FragFlow. First, a data preparation step is necessary to filter and format the workflow corpus. Then the chosen graph mining algorithm is applied, and the results filtered. Finally the results are visualized, statistics are calculated, and the fragments are linked to the workflows of the corpus where fragments appear. The following subsections explain in detail the preparation, filtering and linking steps.

A. Data Preparation

In this step workflows are converted to LDAG format. For each LDAG, the label of each node in the graph corresponds to the type of the step in the workflow, while the edges capture the dependencies between different steps in the workflow. Duplicated workflows are removed from the input corpus. For example, if the source of the corpus is the monthly executions in a server, it is likely to have many repeated workflows. In these cases, the most common fragments would be the most frequently executed workflows, what would be wrong.

Single step workflows are also removed, as they are not meaningful fragments. Single step workflows are often run to test the function of a component of the library, which is then plugged into a bigger workflow.

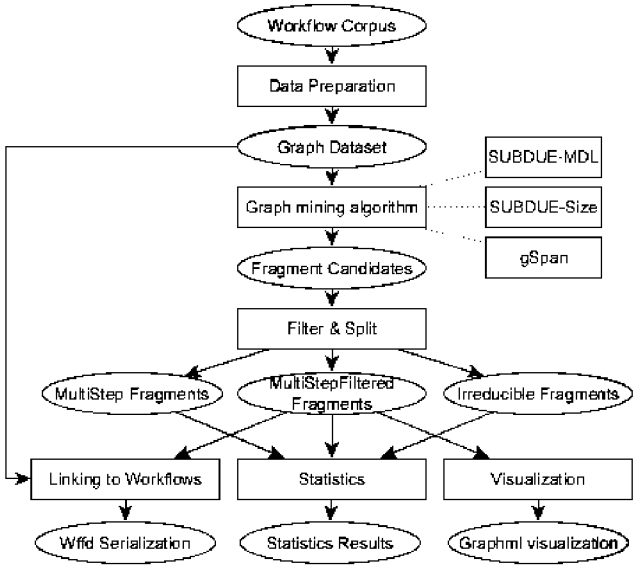

Figure 2: An overview of FragFlow. The rectangles represent major steps, while the ellipses are the inputs and results from each step. Arrows represent where an input is used or produced by a step.

B. Filtering and splitting workflow fragments

FGM algorithms might return fragments that overlap or are included in other fragments. Thus we filter them to present refined results. We distinguish two types of fragments:

a) Multistep fragments: fragments that are composed of more than one step.

b) Multistep filtered fragments: multistep fragments that include other smaller fragments with the same number of occurrences as the larger fragment. Figure 3 shows an example, where four fragments are represented (f1, f2, f3 and f4). Two fragments are part of other fragments (f1 is included in f2 and f3 in f4). However, the frequency of f1 is equal to f2, while f3 occurs more times than f4. Therefore, f2, f3 and f4 are considered multistep filtered fragments, while f1 is not.

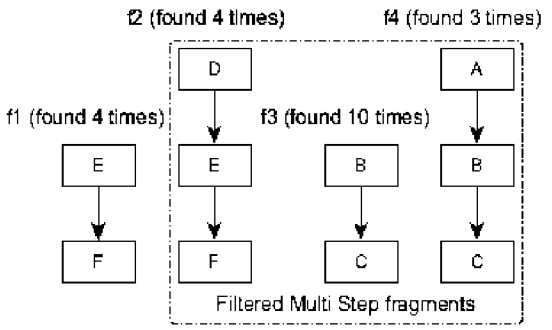

Figure 3: Multistep fragments versus Filtered Multistep fragments

In this work, we use multistep filtered fragments to simplify the results of exact FGM techniques, as they reduce drastically the number of fragment candidates. Other types of fragments are kept for statistical reports. This includes irreducible fragments (i.e., fragments that do not contain other fragments), irreducible multistep fragments (irreducible fragments with at least two steps), discarded fragments (fragments not considered as multistep filtered fragments,) etc.

C. Linking Fragments to Workflows

We bind each fragment to the original workflow corpus, in order to be able to relate different workflows with their common fragments. The rest of this section describes the model we use to represent and bind fragments to results and how the different fragments are bound to the workflows.

1) The Workflow Fragment Description Ontology

The Workflow Fragment Description Ontology[2] (Wf-fd) aims to represent workflow fragments and link them to their occurrences in a workflow dataset. As shown in Figure 4, Wf-fd extends the p-plan ontology[3] [7], as workflow fragments are always part of a workflow specification (p-plan:Plan). In particular, a wffd:WorkflowFragment is a subclass of p-plan:Plan. A workflow fragment has steps (reusing p-plan:Step) which represent the individual data manipulation steps of a particular fragment. The order among the steps is also captured with the property p-plan:isPrecededBy between fragment steps.

In Wf-fd there are two types of wffd:WorkflowFragments. On the one hand, a wffd:DetectedResultWorkflowFragment is a type of workflow fragment candidate, found after applying graph mining techniques or manual analysis on a workflow dataset. It identifies a unique fragment that can be found in a workflow dataset. On the other hand, a wffd:TiedWorkflowFragment represents how a wffd:DetectedResultWorkflowFragment was found in the workflow dataset, pointing to the particular steps of the workflows that are part of the fragment. An example is represented in Figure 5, where a fragment (resultF2) is found one time (linked through the tiedF4 detected result fragment) in a workflow (Workflow 2).

This separation allows each fragment to point to the specific steps of the workflow where it was found (wffd:foundAs), enabling queries to retrieve additional metadata for each fragment (e.g., number of times that a fragment has been detected in a workflow, how the fragment was found, etc.).

Workflow fragments may be included as part of other workflow fragments. To capture this relationship among the detected result workflow fragments, we use the relationship wffd:isPartOfWorkflowFragment.

2) Finding fragments in workflows

Once the final set of fragments has been obtained, we link them to the original workflow corpus represented in Wf-fd. To find where a fragment occurs in the original workflow corpus, we create SPARQL queries with each fragment.

[2] http://purl.org/net/wf-fd
[3] http://purl.org/net/p-plan

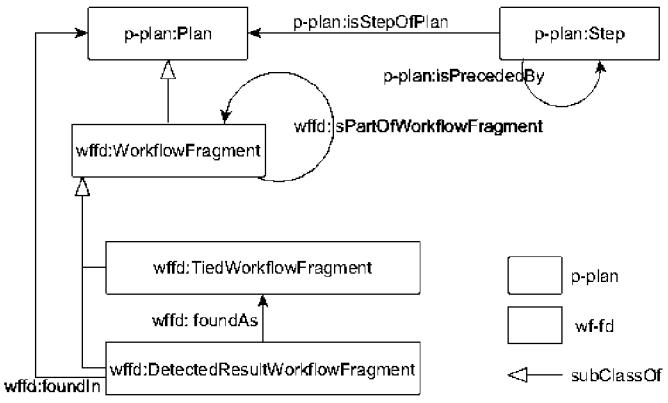

Figure 4: Wf-fd overview

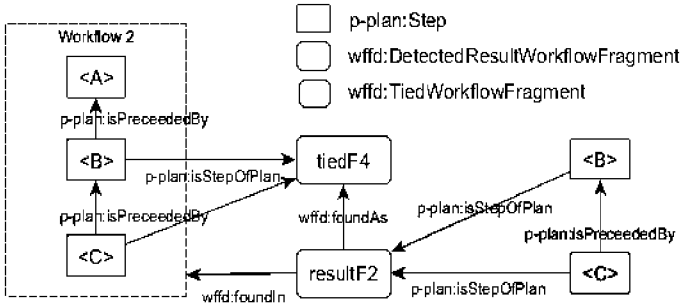

Figure 5: Wf-fd example. A fragment (resultF2) is found one time (tiedF4) in workflow 2.

Workflow fragments detected by exact FGM techniques are trivial to transform to SPARQL queries: each node is transformed to a p-plan:Step, and each dependency among two steps is represented with the p-plan:isPreceededBy relationship. However, workflow fragments detected with inexact matching techniques are more complex to transform. For example, consider the fragment represented on the left of Figure 6 (Fragment 1), where step A is followed by Fragment 2 (composed by two steps B and C). Fragment 1 determines that step A is followed by Fragment 2, but it doesn't specify if step B or C (due to the inexact matching approach). Therefore, we try both possibilities, translating the fragment to the two possible interpretations shown on the right of the figure (Query 1 and Query 2).

The answers for each query (or set of queries in inexact matching) of a fragment are all the possible bindings within all workflows, showing how and where each fragment was found in each of the workflows of the original corpus.

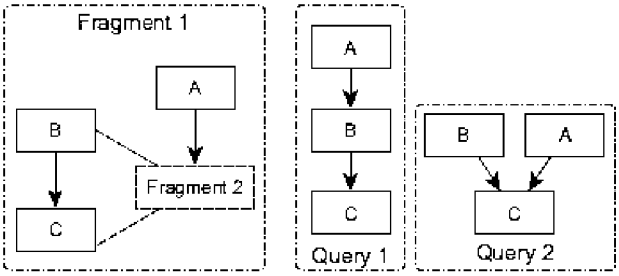

Figure 6: Transforming inexact FGM fragments to queries. The fragment of the left can be transformed to two different queries (inexact approach).

V. EXPERIMENTAL SETUP

This section describes the main features of the corpora that we use to evaluate our approach (Section V.A), along with the types of users who have developed it, and defines the metrics to evaluate our approach (Section V.B). All the corpora, evaluation and results used in this paper are available online as a Research Object[4].

A. Workflow Corpora

We tested our approach on three different workflow corpora created with the LONI Pipeline workflow system (described in Section II). Two features of the LONI Pipeline are of interest for this work: 1) it enables users to define groupings in workflows and 2) it exposes a library of components identified in a unique way with well defined functionality[5], which allows users to reuse popular components. We obtained two corpora from two different users, containing all the workflows created by them or in collaboration with other people. A third corpus contains the runs of 62 unique users submitted to the LONI Pipeline servers during January 2014.

1) User Corpus 1 (WC1)

A set of 790 workflows (475 workflows after applying our filtering) designed mostly by a single user. Some of the workflows are product of collaborations with other users, which produced different versions of workflows originally produced by this user. The domain of the workflows is in general medical imaging (brain image understanding, 3D skull imaging, genetic modeling of the face, etc.), and some are still used by the LONI Pipeline community. Other workflows were designed for a specific purpose and are not reused anymore.

2) User Corpus 2 (WC2)

A set of 113 well-documented workflows (96 after filtering) created and validated by one user, sometimes in collaboration with others. Most of the workflows have been made public for others to reuse[6], and range from neuroimaging to genomics. Some of the workflows were developed as early as 2007, and many of them are being used in different institutions.

3) Multi-user Corpus 3 (WC3)

A set of 5859 workflows (357 after filtering), submitted to LONI pipeline for execution by 62 different users over the time lapse of a month (Jan 2014). The number of filtered workflows descends drastically from the input corpus as many of the executions are on the same workflow or are one component workflows designed for testing.

TABLE 1: CORPUS OVERVIEW

	Corpus	*Original size*	*Size after Filtering*
Single user	WC1	790	475
	WC2	113	96
Multi user	WC3	5859	357

[4] http://purl.org/net/escience2014
[5] http://pipeline.loni.usc.edu/explore/library-navigator/
[6] http://wiki.loni.usc.edu/twiki/bin/view/CCB/PipelineWorkflows

In all three corpora, workflows are likely to reuse components from the public library, what allows groupings to be reused across different workflows. Table 1 provides an overview of the size of each corpus before and after filtering.

B. Evaluation metrics

This subsection introduces the metrics to evaluate our approach with respect to our first two goals defined in Section II. The third goal has been assessed by analyzing the distribution of the groupings in the corpora, so no additional metrics have been defined.

1) Are automatically detected workflow fragments similar to user-defined groupings?

We use precision and recall to measure how FragFlow fragments correspond to LONI Pipeline groupings and workflows. We also consider LONI Pipeline workflows because some of the FragFlow fragments correspond to workflows defined by users. We use precision and recall metrics, defined as:

$$P(\text{Exact}) = \frac{|\text{FragFlow Frag} \cap (\text{LONI Gr.} \cup \text{LONI wfs})|}{|\text{FragFlow Frag}|}$$

$$R(\text{Exact}) = \frac{|\text{FragFlow Frag} \cap (\text{LONI Gr.} \cup \text{LONI wfs})|}{|(\text{LONI Gr.} \cup \text{LONI wfs})|}$$

In the formula, the intersection between FragFlow fragments and the union of all LONI Pipeline groupings and workflows is calculated by measuring which FragFlow fragments are equal to a grouping or a workflow.

We also relaxed the previous measure to look for those fragments that have a significant overlap (more than 80 %) of their steps with a grouping or workflow defined by the user (P(overlap) and R(Overlap) respectively). This additional measure determines how similar our fragments are with respect to a user defined grouping.

The recall includes all groupings and workflows of the dataset on the denominator and those are not necessarily reused, thus we expect the recall to be very low. Also, as we do not know a priori how many of these groupings or workflows are commonly reused, we cannot remove them from the metric.

2) For those automatically detected workflow fragments that are not similar to user defined groupings, do users find them useful?

To assess this goal, we perform a user evaluation, asking users to assess whether candidate FragFlow fragments are acceptable as groupings or not. We measured:

$$\text{Accuracy} = \frac{|\text{FragFlow Frag accepable groupings by user}|}{|\text{FragFlow Frag}|}$$

Similar to what happened for the previous goal, we also assess if the users would have first modified small parts of the fragment before adopting it or not.

VI. EVALUATION

This section presents the results of the evaluation of FragFlow, based on the metrics defined in subsection V.B. First, a quantitative evaluation of the inexact and exact FGM techniques is introduced in subsection VI.A. Then a preliminary user evaluation is described in subsection VI.B;

while subsection VI.C discusses some observations about the usage of workflow groupings in the dataset. A summary of our findings can be seen in subsection VI.D.

A. *Quantitative evaluation*

We applied both inexact techniques (MDL, Size) and exact FGM techniques (gSpan) to test our approach. We decided to leave FSG (exact FGM) out of the evaluation, as it is designed to work with undirected graphs and may return incorrect fragments. The results of the evaluation are described below.

1) Inexact FGM

Table 2 shows the details of the results obtained by applying the MDL and Size heuristics of the SUBDUE algorithm to the different corpora. Both of these techniques are frequency-based approaches, that is, the frequency represents the number of times a fragment is found in the corpus (counting several times if the fragment appears several times in one workflow). The fragment frequencies are normalized according to the size of the dataset, in order to show how different frequencies affect the precision and recall of the found FragFlow fragments. The "min" row stands for the minimum frequency for a fragment to be detected, i.e., it appears at least two times in the corpus.

For all three corpora many workflow fragments are found to be commonly reused. However, there are more common fragments with high frequency in the first two corpora than in the third (where there are no fragments with frequency higher than 10% of the size of the dataset). This can be attributed to the high number of users contributing to the corpus, while the first two corpora had a reduced set of contributors.

In general, the maximum precision obtained ranges from 40% to 75% in the corpora (approx), increasing to 72% to 80% (approx) when we consider the overlap approach. For the first two corpora the higher the frequency of a fragment is, the higher it is likely to be similar to a user defined grouping (with an overlap in their steps higher than 80%). In the third corpus this doesn't hold (10% of precision loss), which can be due to the heterogeneity of the users designing the corpus.

As expected, in all three corpora the recall is very low (32% for the best case, and usually under 20%), as all LONI groupings and workflows defined by users (whether reused or not) are included in the metric. When the frequency increases the recall decreases (down to 0,1% in some cases), according to the number of fragments found.

2) Exact FGM

Table 3 shows the results of applying the exact FGM techniques (gSpan algorithm) to the three corpora. In this case the techniques applied are transaction based, which means that the frequency percentage shown on each row represents the percentage of workflows in the corpora where a FragFlow fragment was found.

As we described in Section III.A, exact FGM techniques aim to find all the possible workflow fragments in the corpus. Thus, when looking for fragments with support less than 5% for corpora 1 and 2, the number of fragments aimed to be returned by the algorithm is so high that the system runs out of memory. For the third corpus, the number of common fragments is lower, so the system manages to return results.

When the fragments appear at least in 10% of the workflows in the dataset, the number of fragments found is still very high compared to inexact FGM techniques. However, after applying our filtering techniques, the FragFlow fragments are drastically reduced. This is because most of the fragment candidates detected by the exact FGM algorithm are not multistep filtered fragments.

A surprising result is that the precision is worse than the fragments found by inexact FGM techniques. This could happen for two main reasons: a) Some of the FragFlow filtered fragments are very small (with two or three steps), and even though they are present in many different workflows, users may not consider them for groupings, and b) Some of the FragFlow filtered fragments have too many steps, which include the main smaller LONI Pipeline groupings designed by users with equal frequency. An example of b) can be seen in Figure 7. On the left of the figure a grouping is defined with a sub-grouping included in it (sub-grouping with steps B and C). Assuming that both the grouping and the sub-grouping are found the same number of times in the corpus, the fragment detected will be the one seen on the right of the figure, thus ignoring the sub-grouping.

B. *User evaluation*

To evaluate whether the FragFlow fragments that do not overlap with LONI groupings would be useful to users or not, we performed a preliminary evaluation by contacting the main contributors to corpus WC1 and WC2. Both users were provided with a set of 16 to 18 randomly selected FragFlow fragments produced for corpus WC1 and WC2 respectively, and were asked if they would use the fragment as it is, they would use it with major or minor changes (i.e., changing more or less than 30% of the fragment), or they would not use it.

The responses are summarized in Table 4. On the one hand, user 1 would select 66% of the proposed fragments (66% accuracy), using 11% as proposed, changing slightly 16% of them and doing major changes to 38% of them. When asked about the reasons to not use 33% of the FragFlow fragments, the user answered that they were too simple (two or three steps).

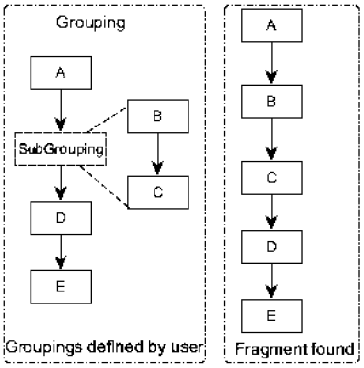

Figure 7: Groupings defined by user versus fragments found. If a user defines sub-groupings that occur with the same frequency as the bigger fragments, then only the outmost fragment will be found by FragFlow.

On the other hand, user 2 would reuse 100% of the fragments detected (100% accuracy), 43% of those as FragFlow detected them, 50% by changing more than one

TABLE 2: INEXACT FGM RESULTS. THE RESULTS ARE NORMALIZED ACCORDING TO THE SIZE OF THE DATASET.

					Exact			*Overlap (>80%)*		
Corpus	***Workflows (w) + groupings(g)***	***Inexact FGM***	***Frequency***	***MultiStep Frag.***	***Fragment***	***Precision***	***Recall***	***Fragment***	***Precision***	***Recall***
WC1	475(w)+ 209(g)	MDL	min	264	76	29%	11%	113	42%	16%
			2%	64	21	32%	3%	27	42%	3%
			5%	26	9	34%	1%	11	42%	1%
			10%	19	8	42%	1%	10	52%	1%
		Size	min	381	136	35%	19%	223	58%	32%
			2%	52	20	38%	2%	32	61%	4%
			5%	22	8	36%	1%	14	63%	3%
			10%	10	3	30%	0,4%	8	80%	1%
WC2	96 (w)+108(g)	MDL	min	95	15	15%	7%	21	22%	10%
			2%	95	15	15%	7%	21	22%	10%
			5%	12	3	25%	1%	3	25%	1%
			10%	5	2	40%	1%	2	40%	1%
		Size	min	88	17	19%	8%	34	38%	16%
			2%	88	17	19%	8%	34	38%	16%
			5%	14	4	28%	2%	9	64%	4%
			10%	4	3	75%	1%	3	75%	1%
WC3	375(w)+ 175(g)	MDL	min	186	100	50%	18%	117	62%	21%
			2%	23	7	30%	1%	11	47%	2%
			5%	4	1	25%	0,1%	2	50%	0,3%
			10%	0	0	0%	0%	0	0%	0%
		Size	min	178	101	56%	18%	119	66%	22%
			2%	22	12	54%	2%	16	72%	3%
			5%	8	3	37%	0,5%	4	50%	0,7%
			10%	0	0	0%	0%	0	0%	0%

TABLE 3: EXACT FGM RESULTS (GSPAN). THE RESULTS ARE NORMALIZED ACCORDING TO THE SIZE OF THE DATASET.

					Exact			*Overlap (>80%)*		
Corpus	***Wf (w) + groups. (g)***	***Support***	***MultiStep Fragments***	***MultiStep Filtered Fragments***	***Fragments***	***Precision***	***Recall***	***Fragments***	***Precision***	***Recall***
WC1	475(w) + 209(g)	5%	Out of memory	-	-	-	-	-	-	-
		10%	51613	16	1	6,2%	0,1%	11	69%	1%
		15%	2264	8	6	75%	0,8%	6	75%	0,8%
		20%	3	1	0	0%	0%	0	0%	0%
WC2	96 (w) + 108(g)	5%	Out of Memory	-	-	-	-	-	-	-
		10%	33236	4	0	0%	0%	1	25%	0,4%
		15%	25	2	0	0%	0%	0	0%	0%
		20%	0	0	0	-	-	0	-	-
WC3	375(w) + 175(g)	5%	5701	3	1	33%	0,1%	1	33%	0,1%
		10%	1074	1	1	100%	0,1%	1	100%	0,1%
		15%	1	1	0	0%	0%	0	0%	0%
		20%	0	0	0	-	-	0	-	-

third of the components, and 6% with minor changes. When asked about the complexity of the fragments, user 2 argued that sometimes additional groupings would be necessary, since they help clarifying and organizing the workflow.

TABLE 4: PRELIMINARY EVALUATION OF FRAGFLOW FRAGMENTS

User	*Use as proposed*	*Use with minor changes*	*Use with major changes*	*Not use*
User1 (WC1)	11%	16,6%	38%	33,3%
User 2 (WC2)	44%	6%	50%	0%

C. Analysis of user-created groupings

Table 5 shows the statistics and distribution of user-defined groupings per corpus. The total number of groupings is more than twice the number of unique groupings, indicating that groupings are reused. Also, groupings are not found alone in corpora workflow. It is common to find more than four groupings in a workflow (when a workflow uses groupings).

The number of workflows with groupings is higher when a single user created the corpus (327 out of 475 and 42 out of 96 from Corpus 1 and 2 versus 89 out of 357 of Corpus 3). In WC1 and WC2 the creators of the workflows are experienced users who know their previous workflows and are likely to reuse them, while the high number of users contributing to WC3 makes it difficult for all of them to be aware of the workflows from other colleagues.

Another interesting fact is the size of the groupings size, being up to 60 in Corpus 3 and down to 0 in Corpus 2. After exploring several workflows showing this practice, we have realized that the high number of steps for some groupings is because users sometimes declare a whole workflow as a grouping. A possible explanation is that this would either facilitate copying and pasting the grouping into other templates, or either help organizing the workflow for the creator: when workflows are too complex, users often separate their functionality in several smaller workflows. Then, each smaller workflow is declared a grouping and copied and linked in a bigger workflow.

Regarding the minimum size of groupings, we have discovered that sometimes workflow creators group unused inputs or outputs in workflows, leading to groupings of 0 steps. A possible explanation to groupings of size 1 is that the workflow creators annotate extra instructions when using a specific component in a workflow (in our analysis only groupings of size >1 are considered, shown in the unique multistep grouping column of Table 5).

TABLE 5: STATISTICS AND DISTRIBUTION OF GROUPINGS IN THE CORPORA

Corpus	*Total qroup.*	*Unique multistep qroup.*	*Wf with qroup.*	*Avg. group. per wf*	*Max n°of steps in qroup.*	*Min n° of steps in qroup.*
WC1	1463	209	327	4	56	1
WC2	302	108	42	7	39	0
WC3	456	175	89	5	60	1

In general, groupings defined by users depend on the granularity that the user is interested in. Some users may not want to see many workflow steps at the same time, and therefore some groupings may contain other sub-groupings, (which may simplify visualization of the workflow), while other users define groupings just according to the functionality of their inner steps.

D. Findings

Our findings are related to the goals presented in Section II. Our first main finding (goal II.a) is that when the fragment frequency is set to 10% of the size of the corpus, 30% to 75% of the total FragFlow fragments found *correspond directly to user-defined groupings in the single user corpora.* In the multi user corpus, *the best results are 50% to 56% with minimum frequency*. If we consider the overlap of 80% of the steps, the precision is 40% to 80%. However, *there is no common configuration* in FragFlow to obtain the best fragment results, as the fragments found depend on how users define the groupings on each corpus.

Our second main finding is that users *find FragFlow fragments proposed as useful candidates for groupings, and therefore useful for reuse in their workflows* (goal II.b). For one user 66% of the proposed fragments were useful, for another 100% were useful. Even though this preliminary user-based evaluation cannot be considered definitive (an evaluation with more participants is needed), it indicates that FragFlow can be useful to users. In this regard, in our case it is better to be accurate with the fragments suggested to the user rather than trying to find all the possible desired groupings, as that could overwhelm the user with suggestions.

Lastly, we studied the distribution and size of the groupings in the corpora, which gives an insight into how workflows and groupings are reused (goal II.c). We showed how likely they are to be reused, and found that there is a minimum of 4 groupings per workflow in those workflows using groupings. We also showed how much they are reused, with 209 groupings reused 1463 times in WC1. Although the analysis in Table 5 gives an idea on how groupings are reused in the corpora, a further analysis studying where each grouping and workflow is reused is necessary (similar to the approach in [20]). This will also help to refine the recall metric for the evaluation of FragFlow.

VII. RELATED WORK

Workflow discovery for reuse has been addressed in prior work. Goderis and colleagues [11] apply sub-graph isomorphism techniques for finding the most similar workflows to a given one, and perform benchmarks for workflow discovery on that basis [10]. Bergman and Gil [2] go a step further, enabling users to find workflows according to different criteria (e.g., having a specific input or output type). While these approaches aim to discover workflows based on certain parameters provided by the user, in our work we aim to expose the most common fragments already in use in a workflow corpus.

Other approaches are based on data mining techniques. Process mining approaches have been used to extract Petri nets and decision trees from event logs [1] [19], in order to know which decision to take at a certain point. The LONI Pipeline has a Workflow Miner[7] module, which follows a statistical approach by measuring the components that most likely precede or are followed by a given component. Leake and Kendall-Morwick [14] rely on CBR approaches to mine provenance traces in order to suggest components when users

[7] http://pipeline.loni.usc.edu/products-services/workflow-miner/

edit new workflows. The difference with our approach in these cases is that we propose the most reused fragments as new workflows/groupings, instead of deriving the whole process network to choose the next most probable component when designing a workflow.

Other approaches also use graph mining techniques to derive the most common usage patterns in a repository of workflows [4], including our previous work [5]. However, they are either limited to a single type of algorithm (SUBDUE MDL [4]), or they provide limited means to refine and link fragments to the input workflow catalogue. In this work we have shown up to three different graph mining algorithms to detect fragments, and we have introduced methods to filter and link them to the original workflows.

Finally, other work [18] explores the application of topic modeling analysis to workflow repositories in order to cluster components and explore them easily. That approach could be combined with our work to allow users to explore proposed FragFlow fragments.

VIII. Conclusions and Future Work

In this paper we introduced an approach to find the most common fragments in a corpus of workflows. We described an implementation of our approach, FragFlow, which integrates several graph mining techniques, and we evaluated the results against user defined groupings of three corpora belonging to the LONI Pipeline system.

FragFlow can be used with different settings, varying the minimum or maximum frequency of the fragments to find, their minimum and maximum size and the type of the graph mining algorithm to be applied. By combining different configurations, we believe we will be able to improve the outcome of our system, according to user-defined preferences.

FragFlow also allows integrating new algorithms as part of its catalog. We are exploring Sigma [16] (inexact FGM) and Gaston [17] (exact FGM) as alternative algorithms.

Future work includes testing FragFlow with other workflow systems (Galaxy [8], Taverna [15], etc.), domains, and further user evaluations. Ultimately, we would like to evaluate how workflows and workflow fragment reuse improve when users are proposed automatically mined workflow fragments.

Acknowledgments

The authors would like to thank Zhizhong Liu, for the help retrieving the workflows, Varun Ratnakar for his technical support and Idafen Santana for his feedback. This research was supported in part by the US National Science Foundation with awards IIS-1344272 and ICER-1343800, by the Spanish Science and Innovation Ministry (MICINN) with an FPU grant (Formación de Profesorado Universitario), by the MEC grant TIN2013-46238-C4-2-R and by NIH grants R01AG040060, R01 NS080655, R01 EB008432, R01MH097268 (to PT) and P41 EB015922 (to AWT).

References

[1] W. M. P. van der Aalst, H. T. de Beer and B. F. van Dongen. Process mining and verification of properties: An approach based on temporall logic. Proceedings of the 2005 Confederated international conference on On the Move to Meaningful Internet Systems. P 130-147. 2005

[2] R. Bergmann and Y. Gil. Similarity assessment and efficient retrieval of semantic workflows. Information Systems Journal, v40 p115-127. 2014

[3] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. Journal of Artifcial Intelligence Research, 1:231-255, 1994.

[4] C. Diamantini, D. Potena and E. Storti. Mining usage patterns from a repository of scientific workflows. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC '12). ACM, New York, NY, USA, 152-157. 2012.

[5] D.Garijo, O.Corcho and Y.Gil. Detecting common scientific workflows using templates and executing provenance. In Proceedings of the seventh international conference on Knowledge capture (K-CAP), 33-40. 2013.

[6] D. Garijo, P. Alper, K. Belhajjame, O. Corcho, et al. Common motifs in scientific workflows: An empirical analysis, Future Generation Computer Systems, Volume 36, Pages 338-351. July 2014.

[7] D. Garijo and Y. Gil Augmenting PROV with Plans in P-PLAN: Scientific Processes as Linked Data. In Proceedings of the 2nd International Workshop on Linked Science 2012, Boston, 2012.

[8] B. Giardine et al. Galaxy: A platform for interactive large-scale genome analysis. Genome Research, 15(10):1451-1455, Oct 2005.

[9] Y. Gil, V. Ratnakar, J. Kim, P. A. González-Calero, et al. Wings: Intelligent workflow-based design of computational experiments. IEEE Intelligent Systems, 26(1):62-72, 2011.

[10] A. Goderis, P. Fisher, A. Gibson, F. Tanoh, et al. Benchmarking Workflow Discovery: A Case Study From Bioinformatics. Concurrency and Computation: Practice and Experience. 00:1-7. 2000.

[11] A. Goderis, P. Li, and C. A. Goble. Workflow discovery: the problem, a case study from e-science and a graph-based solution. In ICWS, pages 312-319, 2006.

[12] C. Jiang, F. Coenen and M Zito. A Survey of Frequent Subgraph Mining Algorithms. The Knowledge Engineering Review, Vol. 28:1, 75–105, 2012.

[13] M. Kuramochi and G. Karypis. An Efficient Algorithm for Discovering Frequent Subgraphs IEEE Trans. Knowl. Data Eng. 16(9): 1038-1051, 2004.

[14] D. Leake and J. Kendall-Morwick. Towards case-based support for e-science workflow generation by mining provenance. In Proceedings of the 9th European conference on Advances in Case-Based Reasoning, ECCBR '08, pages 269-283, Berlin, Heidelberg, 2008.

[15] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, et al. Taverna, reloaded. In 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany, 2010.

[16] M. Mongiovì, R. D Natale, R. Giugno, A. Pulvirenti and A. Ferro, R.Sharan. SIGMA: a set-cover-based inexact graph matching algorithm. Journal of Bioinformatics and Computational Biology; 8(2):199-218. 2010.

[17] S. Nijssen and J.N. Kok. A Quickstart in Frequent Structure Mining can Make a Difference, In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 647–652. 2004

[18] J. Stoyanovich, B. Taskar and S. Davidson. Exploring repositories of scientific workflows. In Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science (Wands '10). ACM, New York, USA, Article 7. 2010.

[19] A. Rozinat and W.M. P. van der Aalst. Decision Mining in ProM. Business Process Management. Lecture Notes in Computer Science Volume 4102, pp 420-425. 2006.

[20] J. Starlinger, S. Cohen-Boulakia and U. Leser. (Re)Use in public scientific workflow repositories. In Proceedings of the 24th international conference on Scientific and Statistical Database Management (SSDBM'12), p361-378. 2012.

[21] F. Torri, A.P. Clark, I.D. Dinov, A. Zamanyan, et al. Next generation sequence analysis and computational genomics using graphical pipeline workflows. Genes, 3:545-575. 2012.

[22] Yan, X. and Han, J.W. gSpan: Graph-based Substructure pattern mining, In Proceedings of International Conference on Data Mining, 721–724. 2002.