Similarity Measures for Scientific Workflows

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Dr. Ing.
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
Humboldt-Universität zu Berlin

von
Dr.med. Dipl.-Inf. Johannes Starlinger

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät:
Prof. Dr. Elmar Kulke

Gutachter:
1. Prof. Dr. Ulf Leser
2. Prof. Dr. Mathias Weske
3. Prof. Dr. Carole Goble

eingereicht am: 16.12.2014
Tag der mündlichen Prüfung: 26.05.2015

Acknowledgements

This thesis would not have made it to submission without the continuous support and encouragement by a number of people.

First of all, I thank my supervisor, Prof. Ulf Leser, for the opportunity to pursue research in his group. He balanced analytical stimulus, scientific guidance, and a reasonable amount of laissez-faire patience with my endeavors just right to provide a fruitful research environment over the past years. I greatly enjoyed working with him and have learnt many a lesson well beyond this thesis. On the same lines, the ongoing exchange of ideas and critical assessment of my work, and the motivation provided by Prof. Sarah Cohen-Boulakia have been invaluable. Thank you so much!

I thank Prof. Susan B. Davidson for providing me with guidance and support, and Prof. Sanjeev Khanna for our inspiring discussions during my time as a visiting scholar at Penn, that brought forward some important aspects of this thesis.

I was lucky to conduct this thesis' research under regular review by the board of supervisors in the SOAMED graduate school, especially Prof. Weske, and my fellow graduate students at SOAMED and WBI in Berlin, and Bryan Brancotte at Université Paris-Sud, who listened and engaged in sometimes eager discussions, and created a very welcoming working environment.

My wife Katrin greatly supported my work not only by tolerating the sometimes quite askew work times, but also by listening to the challenges I faced and providing a different and often very helpful perspective to them. Last but not least, my parents always supported me in persuing my academic career. I can not thank them enough.

The DFG RTG SOAMED and DAAD grants D1240894 and 55988515 provided funding for the research presented in the following. Thank you very much!

Abstract

Over the last decade, scientific workflows have been gaining an increasing amount of attention as a valuable tool for scientists to create reproducible in-silico experiments. For design and execution of such workflows, scientific workflow management systems have been developed, such as Taverna, Kepler, Galaxy, and several others. These enable the user to (often visually) create pipelines of tasks to be carried out on the data, including local scripts and applications and web-service calls. Yet, creating scientific workflows is still a laborious task and complex enough to prevent many non computer-savvy researchers from using these tools. As a consequence, there has recently been growing interest in sharing, reusing and repurposing existing workflows. This is reflected by the emergence of online repositories for scientific workflows, which allow workflows to be uploaded, searched, and downloaded by the scientific community. With increasing size of such repositories, methods to compare the scientific workflows stored in them regarding their functional similarity become a necessity. For instance, to allow duplicate detection, similarity search, or grouping of workflows into functional clusters, similarity measures for scientific workflows are an essential prerequisite.

This thesis investigates similarity measures for scientific workflows. We carry out four consecutive research tasks: First, we closely investigate the relevant properties of scientific workflows and their components regarding their similarity and identify characteristics of their re-use. Second, we review and dissect existing approaches to scientific workflow comparison into a defined set of subtasks necessary in the process of workflow comparison, and re-implement previous approaches to each subtask. We create a large gold-standard corpus of expert-ratings on workflow similarity, with more than 2400 ratings provided for 485 pairs of workflows by 15 workflow experts from 6 institutions. For the first time, this allows comprehensive, comparative evaluation of different scientific workflow similarity measures, confirming some previous findings, but rejecting others. Third, we propose and evaluate a novel method for scientific workflow comparison. We show that this novel method provides results of both higher quality and higher consistency than previous approaches, and can easily be stacked and ensembled with other approaches for still better performance and higher speed. Fourth, we show how our findings from the previous steps can be leveraged to implement a search engine using off-the-shelf tools. This search engine performs fast, high quality similarity search for scientific workflows at repository-scale, addressing a premier area of application for similarity measures for scientific workflows.

Zusammenfassung

In Laufe der letzten zehn Jahre haben Scientific Workflows als nützliches Werkzeug für Wissenschaftler zur Erstellung von reproduzierbaren in-silico Experimenten in zunehmendem Maße an Aufmerksamkeit gewonnen. Für die Erstellung, Ausführung und Verwaltung solcher Workflows wurden spezielle Systeme entwickelt wie etwa Taverna, Kepler, Galaxy und einige andere. Diese ermöglichen es dem Nutzer, (oft visuell) Piplines von datenverarbeitenden Aktivitäten zu erstellen, die sowohl lokale Skripte und Anwendungen, als auch Aufrufe von Web-Services beinhalten können. Trotzdem ist das Erstellen eines Scientific Workflows aufwendig und hinreichend komplex, um computerunerfahrene Nutzer am Gebrauch dieser Systeme zu hindern. In jüngerer Zeit wächst daher das Interesse daran, bereits existierende Workflows weiterzugeben, wiederzuverwenden und zu modifizieren. Dies drückt sich im Entstehen von Online-Bibliotheken, sogenannten Repositories, aus, die speziell dafür entwickelt wurden, Workflows zu veröffentlichen, zu suchen, und herunterzuladen. Mit zunehmender Größe dieser Repositories werden Methoden zur Notwendigkeit, die einen Verleich von Scientific Workflows hinsichtlich ihrer funtionalen Ähnlichkeit erlauben. Um etwa die Erkennung von Duplikaten, Ähnlichkeitssuche oder die Gruppierung von Workflows in funktionale Cluster zu ermöglichen, sind Ähnlichkeitsmaße für Scientific Workflows eine unabdingbare Voraussetzung.

Die vorliegende Arbeit untersucht Ähnlichkeitsmaße für Scientific Workflows. Wir leisten dabei vier aufeinanderfolgende Forschungsbeiträge: Als erstes untersuchen wir eingehend ähnlichkeitsrelevante Eigenschaften von Scientific Workflows und ihrer Komponenten und identifizieren Charakteristika für die Wiederverwendung dieser Komponenten über mehrere Workflows hinweg. Als zweites analysieren wir existierende Lösungen für das Vergleichen von Scientific Workflows, zerlegen diese in eine definierte Anzahl von Unterschritten, die für das Vergleichen von Workflows notwendig sind, und reimplementieren existierende Anzätze für jeden dieser Unterschritte. Wir erstellen einen großen Gold-Standard Corpus mit von Experten bewerteten Workflowähnlichkeiten, der über 2400 Bewertungen für 485 Workflowpaare enthält, die von 15 Workflowexperten aus 6 Institutionen beigetragen wurden. Zum ersten Mal erlauben diese Vorarbeiten eine umfassende, vergleichende Evaluation verschiedener Ähnlichkeitsmaße für Scientific Workflows, in der wir einige vorige Ergebnisse bestätigen, andere aber revidieren. Als drittes stellen wir ein neue Methode für das Vergleichen von Scientific Workflows vor. Unsere Evaluation zeigt, dass diese neue Methode sowohl bessere als auch konsistentere Ergebnisse liefert und darüber hinaus leicht mit anderen Ansätzen kombiniert werden kann, um eine weitere Qualitätssteigerung bei gleichzeitiger Beschleunigung zu erreichen. Als viertes zweigen wir, wie die Resultate aus den vorangegangenen Schritten genutzt werden können, um mit Hilfe von Standardkomponenten eine Suchmaschine zu implementieren. Diese Suchmaschine erlaubt schnelle, qualitativ hochwertige Ähnlichkeitssuche im Repositorymaßstab und adressiert damit ein vorrangiges Anwendungsfeld von Ähnlichkeitsmaßen für Scientific Workflows.

Contents

1 Introduction

1.1 Motivation

Over the last decades, the role of computers in natural sciences' research has steadily grown, and has shifted from a merely supporting to a fundamentally necessary one. This is reflected by the notion of *in-silico* experiments: computer-aided methods of data processing have become a major part of the experimental setup in many sciences [48]. Traditionally, these methods have been implemented in the form of custom scripts and applications, created by computer-savvy scientists to cope with their specific data processing needs. Such scripts are usually hard to maintain and reuse, which hinders both reproducibility and sharing of methods. Also, the hurdle of learning - and mastering - a programming language to conduct research, is counterproductive to scientific advances in many fields.

To overcome these obstacles, *scientific workflows* have been introduced [7]. The term *scientific workflow* refers to the intended target audience of such workflows, the scientists, rather than these workflows being an interesting research object in their own right, or even a possible requirement of being a scientist to create such workflows. Scientific workflows strive to replace the legacy of scripting and command line based approaches to data extraction, processing, and analysis currently still prevalent in many fields of data-intensive scientific research by enabling the user to declaratively, and often visually create pipelines of tasks to be carried out on the data, including both local scripts and applications, and web-service calls. As such, scientific workflows have gained an increasing amount of attention as a valuable tool for scientists to create reproducible in-silico experiments [41] and, today, are used in a variety of domains, including biology, chemistry, geosciences, medicine, or physics [41]. Recently, interest in scientific workflows has further been fueled by the incent of new scientific paradigms: Both processing of very large scientific datasets, i.e., 'Big Data', and sharing of computational methods to ease scientific collaboration in the age of e-Science, are forthcoming necessities scientific workflows lend themselves to [91].

For design and execution of scientific workflows, specialized applications, termed scientific workflow management systems (SWFM), are available. Examples of these SWFMs include Taverna [68], Kepler [14], VisTrails [38], or Galaxy [45]. Yet, while easier than traditional programming, creating scientific workflows using an SWFM is still a laborious task and complex enough to prevent many non computer-savvy researchers from using these tools [90]. Especially for the primary target audience of scientific workflows, the scientists, this hurdle is often still too high. As a con-

sequence, the increasing use of scientific workflows themselves has come with great interest in sharing, reusing and repurposing such workflows: Once designed, a scientific workflow is a valuable piece of knowledge, encoding a (usually proven) method for analyzing a given data set (much like an *executable* Methods part of a paper). Sharing such workflows between different authors and research groups is a natural next step towards increasing collaboration in research which may have a large impact on research, in the same manner as the increased sharing of data sets across the Internet was important for advancing science [10]. This is reflected by the emergence of online repositories for scientific workflows, such as CrowdLabs [64], SHIWA [3], the repositories offered with Kepler [14] and Galaxy [45], or myExperiment [74], currently containing more than 2500 workflows from various disciplines. Such repositories, together with the increasing number of workflows uploaded to them, raise several new research questions to address the challenges of managing large collections of scientific workflows and for using them as a source of expert-supplied knowledge [21, 44].

One such question is how to best enable both manual and automatic discovery of the workflows in a repository to support the user in finding the appropriate workflow for a given task or dataset. Here, challenges include the detection of functionally equivalent workflows, grouping of workflows into functional clusters, the use of existing workflows in the design of novel workflows, or similarity-based workflow retrieval [82, 78, 76, 8, 43]. For instance, arranging workflows into (possibly hierarchical) clusters of equivalent or highly similar functionality, would enable the user to browse the repository along functional categories to discover workflows of interest. Similarly, applying such clustering to the - sometimes extensive - lists of results retrieved from large repositories in response to an initial, keyword driven search for suitable workflows, could greatly benefit the user in selecting the best workflow from amongst those results. Or, given a workflow they have used before, similar (or complementary) workflows could be suggested which would be instantly executable on the data at hand. The ultimate goal is to allow scientists to use scientific workflows without detailed knowledge of the process of their creation, by providing them with optimal methods of searching - and finding - relevant workflows.

The core operation necessary for meeting any of these challenges of workflow discovery and enabling their use cases is the algorithmic comparison of two workflows regarding their functional similarity, i.e., similarity measures for scientific workflows [21]. Only when algorithms of sufficient quality are available to automatically assess whether two workflows do the same, similar or even related things, can these workflows be clustered or recommended. Such assessment is inherently difficult: On the one hand, scientific workflows in public repositories can be richly annotated, and such annotations may be used to assess their function and similarity. Yet, quality and availability of annotations vary greatly between repositories and, more generically, may not be available on a sample workflow a user is searching with. Scientific workflows themselves, on the other hand, represent graphs, and an enormous body

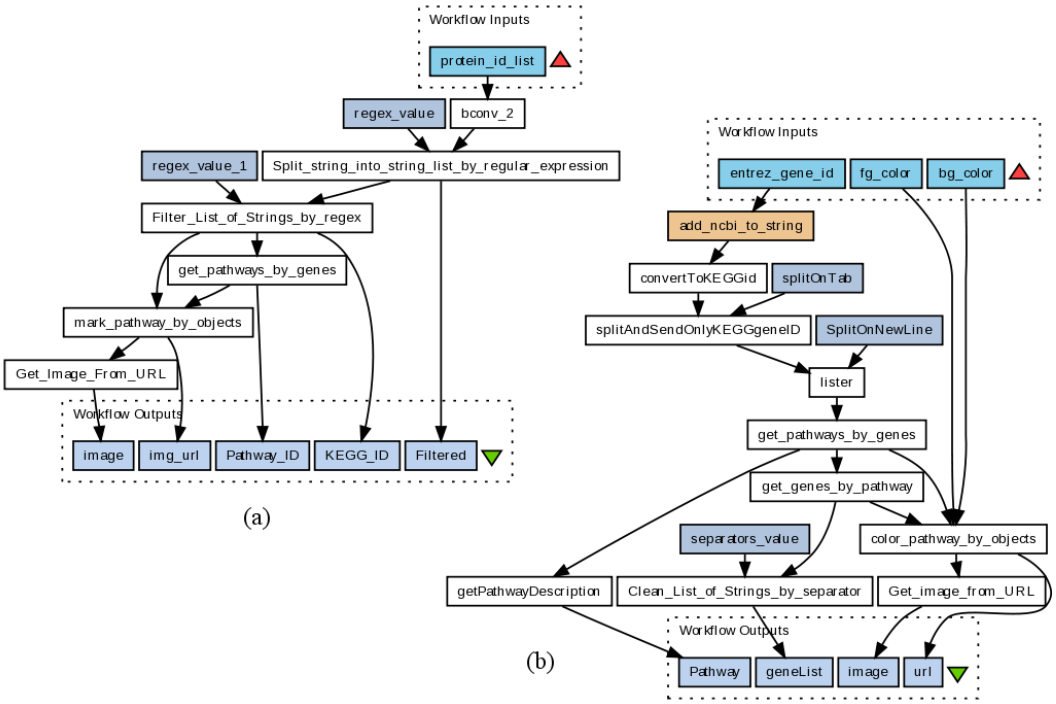


Figure 1.1: Two scientific workflows from myExperiment: (a) ID: 1189, Title: *KEGG pathway analysis*, (b) ID: 2805, Title: *Get Pathway-Genes by Entrez gene id*.

of research exists on graph comparison. Yet, it is not clear to what extent functional similarity of workflows is linked to them being structurally similar. The two workflows shown in Figure 1.1, for instance, arguably offer similar functionality (providing information on relevant metabolic pathways for an input gene or protein), but their structures exhibit a number of differences. Furthermore, each node of such a workflow graph is a distinct data processing task and thus represents a distinct functional element in its own right. From the workflow perspective, each such element is a blackbox, only exposing a limited set of information for assessment of its functionality. How to best identify and compare elements between workflows given this information is still an open question.

This thesis investigates similarity measures for scientific workflows and develops algorithms that deliver high quality results in functional comparison of scientific workflows. In the following sections, we give an overview of our specific contributions and achievements, followed by an outline of the thesis' structure, and an account of own prior work in preparation of this thesis.

1.2 Contribution

The central goal of this thesis is to study methods for the automatic assessment of similarity between scientific workflows, especially in the context of large scientific workflow repositories. In summary, we start at a detailed analysis of workflows and their components in the to-date largest such repository, and continue with an evaluation of existing approaches to scientific workflow comparison. Based on our findings, we propose a novel approach to structure-based comparison of scientific workflows which outperforms previous approaches, and show how it can be effectively stacked and ensembled with other approaches and state-of-the-art information retrieval technology to achieve fast, high quality results.

We make the following main contributions:

Re-use characteristics of workflows and their elements

To compare scientific workflows, the first step is to investigate which elements are shared between workflows, and how these elements can be identified and characterized. Looking closely at almost 900 workflows contained in the largest public scientific workflow repository to-date, we make a number of contributions:

- We explore different methods to identify workflows and their components and determine the most suitable method for detecting re-use.
- We characterize workflow elements not only by the quantity, but especially by the quality of their re-use. Following a functional categorization of workflow elements, we are able to distinguish trivial elements from functionally specific ones. We show that the appearance of single workflow elements in multiple workflows is not per se an indication of functional similarity, and that not all elements are equally well suited for deriving information about functional workflow similarity.
- For the first time, we consider not only sharing of elements across workflows, but also between authors. This allows us to observe that while overall 36% of workflow elements are re-used, only 11% are used by more than one author. This finding greatly supports our motivation of improving workflow discovery through respective similarity measures.

A Framework and Gold-Standard Corpus for Benchmarking Scientific Workflow Similarity Measures

The process of (especially structure-based) comparison of scientific workflows entails several steps. These range from comparison of single workflow elements, to the assessment of the overall similarity value for two entire workflows. Approaches to scientific workflow comparison proposed in previous work have addressed these steps rather heterogeneously and sometimes only implicitly, focusing on the most

prominent aspect of workflow topology. This heterogeneity greatly hinders comparative evaluation of different methods. Additionally, previous evaluations have been carried out either by manual inspection of an algorithm's output, on small corpora which are not publicly available, or by mere comparison to a baseline derived from another computational approach – without consideration of human judgment. A major goal of this thesis is to allow full systematic comparison of different methods for which we make the following specific contributions:

- We carefully analyze previous approaches to measuring scientific workflow similarity and identify a set of necessary steps in the process of workflow comparison. Dissecting previous work along this process, we comprehensively enumerate different approaches taken at each step.

- We set up a framework implementing the workflow comparison process and various methods for each step as adopted from previous work. This framework allows to separate and combine each such method at each step for thorough evaluation. In the framework, we further extend the process of workflow comparison to (optionally) include external knowledge derived from a scientific workflow repository to investigate how such knowledge may benefit workflow comparison quality. Altogether, 288 different setups for the process as a whole are available.

- We collect the to date largest corpus of human workflow similarity ratings, contributed by 15 workflow experts from 6 institutions in 4 countries, which contains more than 2400 similarity ratings for 485 pairs of scientific workflows over a repository of almost 1500 workflows from the Taverna scientific workflow management system. The corpus consists of two sets of ratings: The first set of 1900 similarity ratings over 24 query workflows and ranked lists of 10 workflows compared to each query workflow is independent of any concrete algorithm. It is thus suitable for evaluation both of large sets of algorithms, and of new algorithms to be developed in the future. The second set of ratings was specifically collected for pairs of workflows from the result lists returned by selected, promising algorithms for similarity search over a given repository to determine retrieval quality. For cross-repository evaluation, the corpus additionally contains 181 ratings over a dataset of 139 workflows from the Galaxy system. We make the whole corpus publicly available.

- For the first time, framework and corpus together allow for a comprehensive and systematic, comparative evaluation of different settings and approaches used for each step of the workflow comparison process. We pinpoint different options at each such step, investigate how they influence the quality of the resulting similarity measures as a whole, and combine different measures into ensembles to further improve result quality. Finally, we trace our findings back to the properties of scientific workflows in a number of online repositories.

Layer Decomposition: a Novel Structure-Based Approach for Scientific Workflow Similarity and its Application to Fast Similarity Search at Repository-Scale

Our observations from the differentiated evaluation of existing approaches to measuring scientific workflow similarity indicate potential for improvement with respect to how the workflows' topology is compared: On the one end, approaches that only consider the data processing tasks contained in the compared workflows, and disregard the way they are connected with each other, deliver results of satisfying quality only when other steps of the comparison process are tuned to match the specific properties of a given repository - which limits their applicability to off-the-shelf deployment. On the other end, comparing workflow topologies by substructures greatly reduces the amount of tuning necessary, but is too slow for application at repository-scale. Targeting this apparent trade-off between speed and stability, we make the following contributions:

- We propose a novel approach to the assessment of scientific workflow similarity. This approach is based on a concise representation of the partial order of a workflow's modules, which we term *Layer Decomposition*. This approach not only provides best results in comparison to previous measures, but requires as little tuning as previously found for substructure-based comparison. At the same time, while naturally still being outrun by fully structure-agnostic algorithms, our approach is much faster than other structure-aware approaches.
- Specifically targeting the use case of similarity search over a large scientific workflow repository, we investigate how multiple different comparison methods can be stacked and ensembled to maximize both speed and quality. We implement a system that uses state-of-the-art document indexing for fast structure-agnostic retrieval of candidate workflows, which are reranked by our *Layer Decomposition* approach. Evaluation shows that this system not only delivers high quality results, but is faster than the fastest standalone approach to structure-based workflow comparison by several orders of magnitude.

1.3 Outline of this Thesis

Chapter 2 introduces the necessary concepts and applications, including scientific workflows and online repositories for publishing and reusing them, and gives an overview of related work in the field of workflow comparison.

Chapter 3 presents our analysis of the workflows contained in the to-date largest publicly accessible scientific workflow repository, myExperiment [74], investigating how workflows and their elements are used and re-used across workflows and their authors, and identifying characteristics of most commonly used elements.

Chapter 4 closely inspects the process of scientific workflow comparison and gives a detailed review of previous work, describing how each aspects of the comparison process has been treated. The collection of a human-annotated corpus of similarity ratings is described, on which a rich set of previous approaches is comprehensively evaluated.

Chapter 5 proposes a novel approach to structure-based assessment of scientific workflow similarity, designed to balance the amount of structural information taken into account against runtime considerations. We observe how this approach follows from our findings in the previous chapters and provide a comparative evaluation of its quality in similarity search, and of its runtime properties. We also explore options for further speedup of repository-scale retrieval.

Chapter 6 makes direct use of these findings, presenting a proof-of-concept implementation of structure-aware similarity search for scientific workflows at repository-scale which we show to provide high quality results at high speed.

Chapter 7 concludes the thesis with a summary of its findings, and an outlook on future work.

1.4 Own Prior Work

Chapter 3 presents an analysis of (re)use of workflows and their elements in a large public repository, which has been published by Starlinger et al [81] with multiple authors. The authors' roles can be assigned as follows: Leser supervised the work. All analysis work was done by Starlinger. Starlinger wrote the the manuscript which was critically revised by Leser and Cohen-Boulakia.
Chapter 4 presents a framework for and evaluation of similarity measures for scientific workflows, which has been published by Starlinger et al [79] with multiple authors. The authors' roles can be assigned as follows: The work was conceived by Leser and Starlinger. Leser and Cohen-Boulakia supervised the work. Brancotte provided methods for consensus ranking of expert ratings. All implementation (apart from consensus ranking) and evaluation was done by Starlinger. Starlinger wrote the manuscript, which was critically revised by Leser and Cohen-Boulakia.
Chapter 5 presents a novel approach to scientific workflow comparison which has been published by Starlinger et al [80] with multiple authors. The authors' roles can be assigned as follows: Davidson, Khanna, and Starlinger conceived the Layer Decomposition idea. The work was supervised by Leser and Davidson. Implementation and evaluation was done by Starlinger. Starlinger wrote to the manuscript, which was critically revised by Leser, Davidson, Khanna, and Cohen-Boulakia.

2 Scientific Workflows and Workflow Similarity

In this chapter, we introduce the core concepts of scientific workflows and review how they compare to other types of workflows, including the well studied area of business workflows and processes. In Section 2.2 we provide an overview of existing systems for the creation, management, and storage of scientific workflows, and target the properties of current scientific workflow repositories. In Section 2.3 we then give a comprehensive account of previous work on workflow comparison both for scientific workflows and for other types of workflows.

2.1 Scientific Workflows

Scientific workflows have been introduced as a means to ease the creation of data processing and analysis pipelines for in-silico experiments [7]. These workflows not only lend themselves to the automated processing of large volumes of data, but also facilitate sharing, reuse, and re-purposing of computational methods. Therefore, it is not surprising that both deployment of, and research in scientific workflows have lately gained additional momentum in the context of the e-Science and Big Data movements. Nowadays, scientific workflows are used in a variety of domains, including physics, chemistry, geosciences and the life sciences, in particular, biology and biomedicine [41]. Prominent examples of scientific workflows thus exist in several of these domains: The *Montage* [11] astrophysics workflow generates mosaic images from telescope surveillance photographs of the sky; *CyberShake* [47] is a workflow to compute a specific geographical site's earthquake hazard estimate from a variety of sources of information; and in biology, workflows for the processing and analysis of *Next Generation Sequencing (NGS)* data are establishing themselves as an invaluable tool for processing the large quantities of input data these experiments produce [86]. An abstract example of such an NGS-workflow is shown in Figure 2.1 where the short nucleotide sequences read by the sequencing machine are automatically aligned and mapped to a given reference genome, to assess information on genetic variations discovered in consecutively executed analysis steps. Note that we contrast the abstract workflow shown in Figure 2.1 to concrete workflows in Section 2.1.1 (page 15).

The overall tasks these three workflows carry out, while being rather different in the intended area of application, share the same anatomy: Scientific workflows follow a modular design, where a variety of tools, or *modules*, are composed to create the workflow's overall functionality. These modules are connected into a DAG-structure

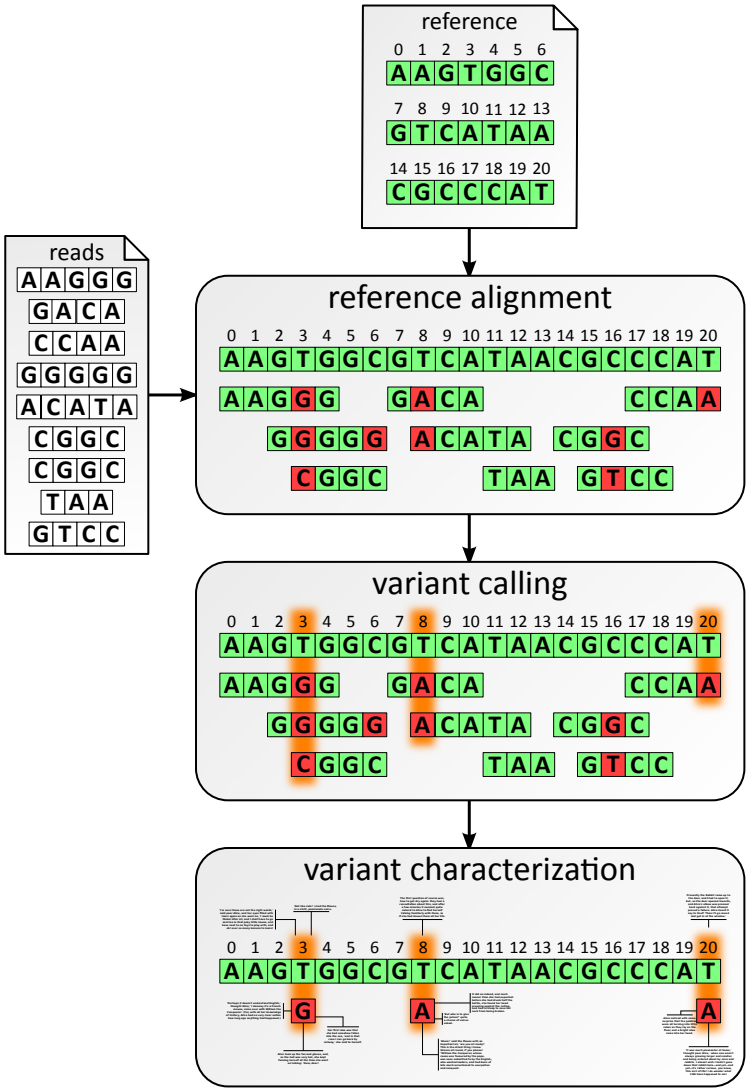


Figure 2.1: Abstract sample workflow for the processing and analysis of experimental data from next generation sequencing. Courtesy of Marc Bux, [17]

by *datalinks*, where each module automatically passes its output to the successive one. While this conceptual anatomy of scientific workflows is rather generic, a vast number of different implementations of this concept exist. These implementations come in the form of so called *Scientific Workflow Management Systems (SWFM)*, each of which has its own set of features, and typically uses its own syntactic format for storing the workflows created in it.

Before we turn to a review of the features of popular SWFM in Section 2.2.1, we here give a more detailed description of the core concepts of scientific workflows common to most of these systems - especially those relevant in the light of this thesis.

We also formally define the notion of scientific workflows considered, and introduce the nomenclature used throughout this thesis. Again, in the broad landscape of scientific workflow systems, not all systems strictly adhere to the concepts described here but provide extensions or adaptions. For instance, it is not uncommon for scientific workflows systems to provide means to the user to explicitly manipulate the underlying flow of execution control within the workflow. We make note of these variations, and provide real world examples from existing SWFM were necessary.

2.1.1 Modules, Datalinks and Dataflows

Data Processing Modules in Scientific Workflows

A scientific workflow consists of a number of distinct subtasks, and each of these subtasks usually represents a full data processing step in its own right. For instance, the subtask of *reference alignment* in the NGS workflow example of Figure 2.1 compares each of the plenitude of short read sequences output by the sequencing machine to a given genome, chromosome, or other nucleotide sequence to act as a reference, and maps it to the most appropriate genetic location. Often, more than one implementation of such a (sub)task exists, e.g., Bowtie [55], BWA [57], or Perm [18] for reference alignment. Such implementations are used as **modules** within a given workflow, and are treated as blackbox components, i.e., no information about their internal working is available.

Technically, each task is usually wrapped into a container to provide a uniform interface within the workflow, regardless of how the task is invoked. Thus, strictly speaking, when we say *module*, we refer to such a container – which is parameterized by the type of operation to carry out (e.g., *web-service call* or *local executable*), and the operation's concrete configuration (e.g., the web-service's url or the executable's path in the filesystem). Which types of operations are available and which configuration parameters these expose, depends on the concrete SWFM, just as much as schema and syntax of their storage in an SWFM-specific workflow file format do. Listings 2.1 and 2.2 contrast the module representation in the *Scufl* workflow format supported by Taverna [68], to the format used by Galaxy [45].

While type and configuration parameters are carefully encoded in the XML schema of the Scufl workflow, containing only the necessary information, the module's definition in the Galaxy workflow contains an extensive set of attributes, including positional information for where the module was laid out in the graphical representation in the SWFM, whereas the specific configuration of the wrapped tool is stringified in the *tool_state* parameter. Despite the apparent heterogeneity, for any given module independent of its type, the configuration most typically includes a descriptive label or *name* reflecting the module's functionality (which is often provided by the workflow author), but it does not include any semantic annotations. To formally capture this diversity, we represent the configuration of a module as a set A of pairs of attributes a and values v:

$$A = \{(a_1, v_1), \ldots, (a_{|A|}, v_{|A|})\}$$

Listing 2.1: Module definition in Taverna's Scufl XML format

```
<s:processor name="getP53MutationsByIds">
  <s:description>
    Get tp53 gene mutations by ids from TP53 IARC database
    (see http://srs.o2i.it/srs71/)
  </s:description>
  <s:soaplabwsdl>
    http://bioinformatics.istge.it:8080/axis/services/o2i.getP53MutationsByIds
  </s:soaplabwsdl>
</s:processor>
```

Listing 2.2: Module definition in Galaxy's workflow format

```
"5": {
    "annotation": "",
    "id": 5,
    "input_connections": {
        "input": {
            "id": 4,
            "output_name": "out_file1"
        }
    },
    "inputs": [],
    "name": "Cut",
    "outputs": [
        {
            "name": "out_file1",
            "type": "tabular"
        }
    ],
    "position": {
        "left": 282.38333129882812,
        "top": 381
    },
    "post_job_actions": {},
    "tool_errors": null,
    "tool_id": "Cut1",
    "tool_state": "{\"columnList\": \"\\\"c1,c5,c5\\\"\", \"input\": \"null\",
                  \"delimiter\": \"\\\"T\\\"\", \"__page__\": 0}",
    "tool_version": "1.0.1",
    "type": "tool",
    "user_outputs": []
}
```

Each attribute thus refers to an SWFM- and type-specific parameter, including aforementioned label (where available), and the type of operation itself.

Each module has input and output ports which interface the tool it wraps to the other modules of the workflow. The attentive reader will have noticed that in the sample Galaxy module from Listing 2.2, these ports are included in the module's definition. For Scufl, on the other hand, a module's ports are only referenced by the datalinks that connect it to other modules (see Listing 2.3). These ports commonly have both a name to identify them, and a syntactic type describing the expected structure of the input, such as *string*, or *tabular*. Semantic typing, or even more fine grained syntactic typing of inputs and outputs, on the other hand, is far less common than would be desirable. The type *string*, for instance, is usually used even for inputs requiring data in specific formats like fasta (used for DNA-Sequences) or XML, or data of a very specific type such as a protein id. We define

$$I = \{(n_{i_1}, t_{i_1}), \ldots, (n_{i_{|I|}}, t_{i_{|I|}})\} \quad \textit{and} \quad O = \{(n_{o_1}, t_{o_1}), \ldots, (n_{o_{|O|}}, t_{o_{|O|}})\}$$

as sets of a module's input ports I and output ports O, where n_{i_l} and n_{o_k} are the ports' names, and t_{i_l} and t_{o_k} are their type.

Altogether, from a syntactic point of view, a module can thus be defined by its set A of configuration attributes and their values, and its sets I and 0 of input and output ports:

$$m = (I, O, A)$$

We refer to specific attributes or ports of a modules as $m : a_i$, $m : n_{i_l}$, and $m : n_{o_k}$, respectively. Note that this definition explicitly does not require a module to carry a distinctive name or label, as not all SWFM provide these. Where available, such names are included in the modules' attributes.

The set of modules used in a workflow is thus

$$M = \{m_1, ..., m_{|M|}\}.$$

Datalinks linking Modules by Data

How different modules are combined and connected in a given workflow is defined by the data they produce or consume: When the output of one subtask serves as input for the successive one, they are connected by a **datalink**. In Figure 2.1, the genetic mapping information from *reference alignment* serves as input for *variant calling* to find differences in the read and the reference sequences - the datalink between these two modules is represented by an arrow connecting them.

Datalinks are typically defined on specific output and input ports of the modules they connect. Listing 2.3, for instance, shows an excerpt of a Scufl workflow linking the input port named *id* of the *getP53MutationsByIds* module from Listing 2.1 to the *filteredlist* output port of its preceding module. Each such port may have multiple datalinks connected to it.

Listing 2.3: Datalink definition in Taverna's Scufl XML format

```
<s:link source="FilterListOfStringsExtractingMatchToARegex:filteredlist"
        sink="getP53MutationsByIds:id" />
```

$$D \subset \{(m : n_{o_k}, m' : n_{i_l}) \mid (m, m') \in M \times M\}$$

is the set of (directed) datalinks of a workflow, where each datalink connects a pair of modules from one module's output to the next module's input port. Note that where a single port has multiple datalinks connected to it, the underlying semantics are always AND: All datalinks sourcing a given output port receive the same data, and module's receiving data on a single input port through multiple datalinks will be executed on each of the inputs in turn. As such, datalinks do not provide branching, and do not carry a state for the data they transport.

Scientific Workflows as Dataflows

The overall functionality of a workflow is thus defined by the entirety of the subtasks it is comprised of (its modules), and how these are orchestrated within the workflow (by datalinks). That is, scientific workflows typically don't explicitly contain elements to control execution of the data processing modules they contain, but represent **dataflows**. Visually speaking, the datalinks define how the data flows from one module to the next through the entire workflow. Once constructed, such workflows are invoked with a set of (user provided) input data and parameters on which they operate in a fully automated manner to produce one or more final outputs which are then returned to the user. The intermediate *reference alignment* data is passed to the step of *variant calling* without user intervention, and execution of modules is triggered by the availability of their respective input data. Note that the term *dataflow* is also frequently used to specifically refer to declarative data processing languages such as Pig [69] or Dryad [50] which follow the same model of computation. Here, we use the term to reflect that this model of computation is used by most SWFM for the scientific workflows created in them [7].

Formally, dataflows represent directed acyclic graphs (DAG), with modules and datalinks as nodes and edges, respectively. A scientific workflow W can thus be defined as

$$W = (M, D)$$

where M and D are the sets of modules and datalinks as defined above. Note that this dataflow-oriented definition explicitly does not distinguish between different classes of modules, such as modules for controlling and manipulating workflow execution, or modules for data processing. Nor does it distinguish between different types of modules within its module set M. As stated above, which types of modules are available depends on a given SWFM. While the respective SWFM, of course,

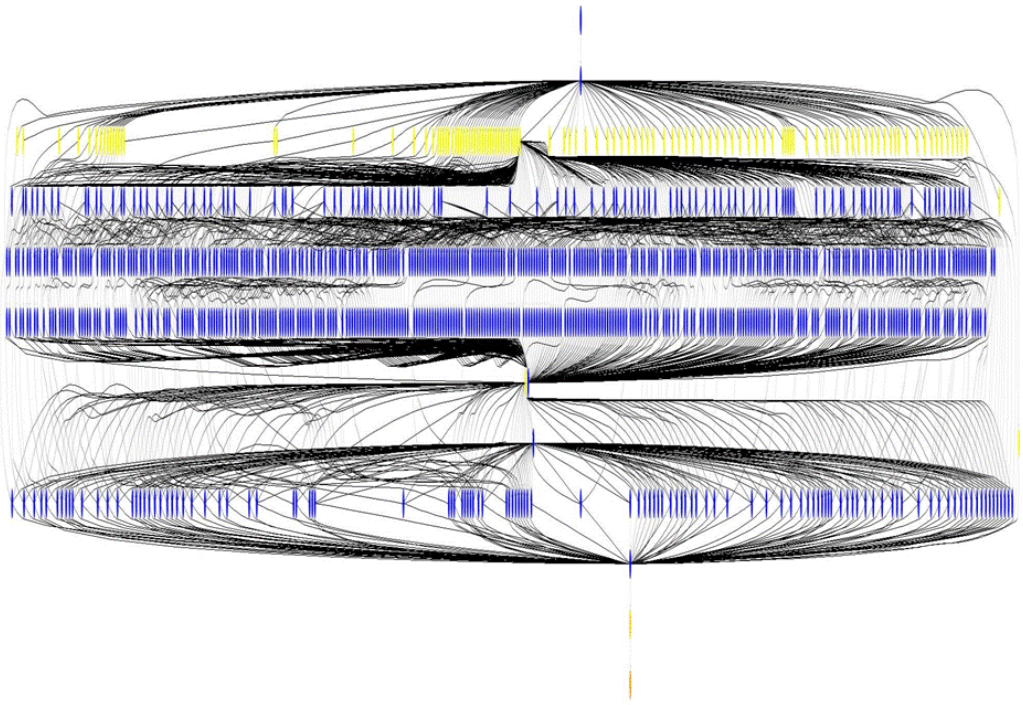


Figure 2.2: Instantiation of a Montage workflow, [52].

distinguishes these types very well, for the sake of generality we, formally, treat each module as a blackbox component. The only exception we make here are modules which contain complete workflows themselves: some SWFM allow to include nested dataflows, or subworkflows, by wrapping these in modules of an appropriate type. For the most part of this thesis, we extract such such nested dataflows from their containing modules and inline their elements with the surrounding top-level dataflow. We make note of exceptions, where this is not the case.

Workflow Definition vs Workflow Instance vs Abstract Workflow

In the scientific literature and discourse, the term *scientific workflow* is frequently used to refer to different representations of these workflows. Reflecting the dataflow paradigm and the absence of explicit control-flow modeling, the workflow defined by the user in an SWFM represents the intent of the computational experiment. The example workflows shown so far depict this user-view on the workflow, i.e., the *workflow definition*. This workflow definition typically specifies which tasks are to be carried out on *all* independent instances of the data passed to the workflow or its modules, without the need for the user to explicitly allocate a dedicated module to each data instance. Such allocation is usually handled by the underlying SWFM,

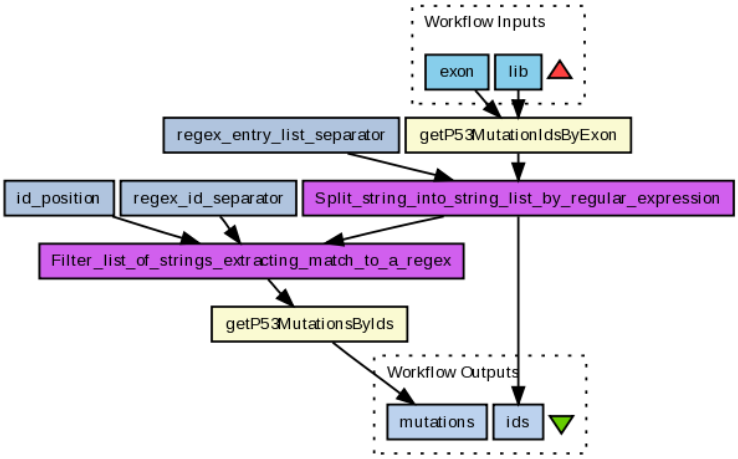


Figure 2.3: Sample concrete workflow definition from the Taverna SWFM to retrieve information about a genetic sequence using several different types of modules (blue: input or string constant; pink: local string operation; beige: web-service call). See also http://www.myexperiment.org/workflows/1013.

before or while the workflow is executed, and inflates the workflow into a concrete *workflow instance*. The amount of inflation typically depends on the number of independent data items present, and the physical resources available for execution. Contrasting our previous examples, Figure 2.2 shows the workflow instance in a specific execution of a Montage workflow for processing telescope images of the sky. Here, each (colored) row holds thousands of instances of a specific module defined for one processing step in the workflow, which is instantiated multiple times for parallel processing of data. Each module instance operates on a separate instance of (intermediate) data. In the corresponding workflow definition, on the other hand, each row will be represented by only a single module.

On the other end of the scale of representation, the intent specified by the user may not even go as far as the *concrete* workflow definition ready for execution, where each module contains information about the concrete task to be carried out and its execution parameters, but is only specified by a description of *what* is to be done (as opposed to the *how*). Such *abstract workflows* again range from rather informal illustrations (such as the one shown in Figure 2.1) to formal plans of which tasks are to be executed, yet without assigning concrete executables to them. It is the responsibility of either the users themselves, or of the SWFM to assign concrete executables or services to each task before workflow execution. For example, in the abstract workflow shown in Figure 2.1, the task of *reference alignment* would thus need to be mapped to a concrete tool, such as Bowtie, BWA, or Shrimp. Making

such assignments automatically, is a line of research in its own right [29, 59], and – depending on the degree of formal planning expressed in a given abstract workflow – may, for instance, require modification of the workflow to include additional steps of data transformation [71]

Throughout the remainder of this thesis, when we use the term *scientific workflow*, we refer to a *concrete workflow definition*, where a workflow's modules are assigned concrete tools to operate on the data. Not only is this the user's view on the workflow most frequently supported by current SWFM, but it is also the representation typically found in scientific workflow repositories. Figure 2.3 shows the concrete workflow from the Taverna SWFM, from which the excerpts from Listings 2.1 and 2.3 were taken.

2.1.2 Other Workflow Concepts

The concept of using distinct interconnected components to form a workflow or process with a composite, overall functionality is, of course, not new or limited to scientific workflows. Here we compare the conceptual and technical aspects of scientific workflows to related technologies. Often enough, there is no clear cut distinction, but only specific aspects of the underlying concepts differ.

Scripted Data Analysis

Scripting languages, such as Perl, Python, or Bash, are often used to glue different external tools together and to implement the necessary data-transformation between one tool's output and the next tool's input. These scripts represent the conceptual predecessor of scientific workflows as they are used in a similar way to pipeline data through a number of consecutive processing tasks. Yet, such scripts are usually rather tedious to create, require the knowledge of (at least one) programming language to write or even understand, and often require substantial effort to be amended when new tools for particular subtasks are to be used, or even slight changes to the overall functionality are required. Additionally, more technical aspects such as parallelized execution of single tasks on multiple independent sets of data, or the inclusion of dedicated code for logging and debugging have to be taken care of explicitly by the script's author - distracting focus from the original experimental intent. Scientific workflows have been introduced to target these issues and simplify the creation of data processing applications, and, in doing so, promise a number of inherent benefits:

- Creation of workflows is (often) visual, and thus more accessible to non-programmers than traditional scripting.
- The dataflow paradigm of scientific workflows abstracts the technical details of execution, allowing the user to focus on the core aspects of the computational experiment. Of course, this abstraction comes at the price of being less

expressive than traditional programming: Turing-complete languages can not be fully resembled by scientific workflows without conditionals or loops.

- Currently SWFM typically provide additional features, such as provenance tracking and execution monitoring - removing the need for the user to include such functionality themselves (see below).

Business Workflows and Processes

Business workflows are a well established concept and research area [85]. In these workflows, compound processes are modeled as a series of relevant activities which are connected in a similar way as the modules in scientific workflows. Even though scientific workflows and business workflows differ already by their intended areas of application – as implied by their names – the actual differences in functionality are not as sharp as one might expect [90, 61]. In the light of the research target of this thesis, the most important distinctions are:

- Scientific workflows are typically dataflow-oriented DAGs whereas business workflows include rich control-flow structures, such as conditionals, branches, and loops. This results in a clear definition of the behavior of a better part of the elements contained in a business workflow, in contrast to the uniform, blackbox view on modules used in scientific workflows.
- The focus of scientific workflows lies in automated data processing. As such, most of their modules represent computational activities. The activities in business workflows, on the other hand, may be either computational or operated by a human. Here, the workflow models how the work is to be distributed to the respective human actors by the system enacting the workflow.
- In contrast to in-company repositories of business workflows, scientific workflow repositories are often *open*, leading to plenty of authors creating workflows, with the result of cross-author differences in workflow design and naming of workflow components.
- The variety of languages scientific workflows are described in is greater than with business workflows, not only, but also due to the absence of a widely accepted standard, such as BPMN or BPEL for business workflows.

Mashups

Mashups were introduced as a means of ad hoc integration of structured data on the web [5]. A typical use case is the cross-source interlinking of (possibly filtered) articles from a news feed with pictures from public libraries of tagged photographs, and their geographical display in an online maps application. Yahoo Pipes [70], for instance, is a system that allows graphical composition of such mashups in a very similar way to how SWFM allow the visual creation of scientific workflows.

While thus very similar to scientific workflows at first sight, a number of important differences exist:

- Mashups almost exclusively work on structured data in the form of RSS, JSON, or related types of web-feeds.
- The intended area of application is specifically that of online data integration - as implied by the term *mashup*. Scientific workflows can be used for data integration, too, but are not limited to this type of usage.
- As opposed to scientific workflows, mashups explicitly allow the modeling of control-flow elements such as loops or conditionals.

Declarative Data Processing Languages

Declarative data processing languages such as Pig [69] or Dryad [50] can be seen as a close relative of scientific workflows. These languages provide high level operators to support analysis processes over distributed infrastructures, abstracting from the underlying computational paradigms (such as Map-Reduce[28]). The differences to scientific workflows are sparse and include:

- The application area of data processing languages is set on data analytics with a focus on relational operations. While scientific workflows can be used for such tasks as well, they don't explicitly support them.
- Declarative data processing languages typically focus on providing predefined operators, e.g., for selecting or joining data. User-defined functions, on the other hand are a corner case. In scientific workflows, every module corresponds to a user-defined function (even when it is provided by the SWFM itself).
- Usually no visual composition is provided with such languages.

Service-oriented Architectures

The concept of service-oriented architectures (SOA) describes the provision of self-contained applications as programmatic services which can be combined in a manual, semi-automated, or automated manner to deliver a composite functionality [37]. Which services a required functionality is provided by is registered with a service broker, that can - in theory - be used to select the best service for a given task [36]. SOA is more of a paradigm than a concrete technology and is orthogonal to the concept of scientific workflows: Scientific workflows can be used as a modeling and execution layer on top of an underlying service-oriented architecture to orchestrate available services. And indeed, many scientific workflows access services (either locally or on the web) to include their specific functionality in the in-silico experiment executed by the workflow. The role of a service broker is typically taken by the workflow's author, who manually chooses the services to include in the workflow.

2.2 Infrastructure for Scientific Workflows

For scientists to make best use of scientific workflows, infrastructure has been developed. This infrastructure targets scientific workflow creation, execution, and storage, and is responsible for providing many of the benefits of scientific workflows to the scientist: For authoring of scientific workflows, *Scientific Workflow Management Systems (SWFM)* are available, which typically also manage workflow execution. For sharing and re-using workflows, *Scientific Workflow Repositories* have emerged. In the following, we introduce this infrastructure and give an overview of the benefits scientific workflows provide through it.

2.2.1 Scientific Workflow Management Systems

For the design and execution of scientific workflows, specialized systems are available, called *Scientific Workflow Management Systems (SWFM)*. Today, numerous SWFM exist, including Taverna, Kepler, VisTrails, Galaxy, Pegasus, Knime, e-BioFlow, e-Science Central, or Askalon (e.g., [68, 14, 38, 45, 31]). These systems act as a central access point for the user to create, run, and monitor the execution of scientific workflows, and sometimes even directly interface with public or local scientific workflow repositories. A comprehensive review of existing systems and their respective capabilities is out of the scope of this thesis. Surveys can be found in [91], [7] or [30]. Here, we give an overview of some of the most prominent features of SWFM.

Graphical Workflow Editing. Most typically, SWFM provide a graphical user interface for accessing their functionality, including the ability to visually assemble and modify workflows. As an example, Figure 2.4 shows the user interface of the Taverna SWFM, where the workflow from Figure 2.3 is opened for editing. Next to the workflow canvas on the right, the lower left pane lists details about the modules used in the workflow. The upper left pane holds the library of available modules.

Libraries of Modules. Most SWFM provide libraries of predefined modules ready to be used in workflows. As many SWFM have defined domains of application, the modules offered often include domain-specific applications, next to generic operations on string, tabular, or XML data. For defining modules not included in the library, wrappers are provided. Depending on the underlying SWFM, various kinds of modules are supported, which may run either locally or on the web. These include command line tools, web-services, or user-provided scripts defined directly within the SWFM. For example, the workflow shown in Figure 2.4 includes web-service calls (beige, grayed out), locally executed string modifiers from the built-in library (pink), and string constants (blue). This great flexibility in including components from a variety of technical pedigrees caters to the frequent need in scientific applications to use both state-of-the-art web-service tools and data, and at the same time include pre-existing local data and scripts, carefully tailored to a specific aspect of the intended analysis, or tools not available as a web-service or inconvenient to use

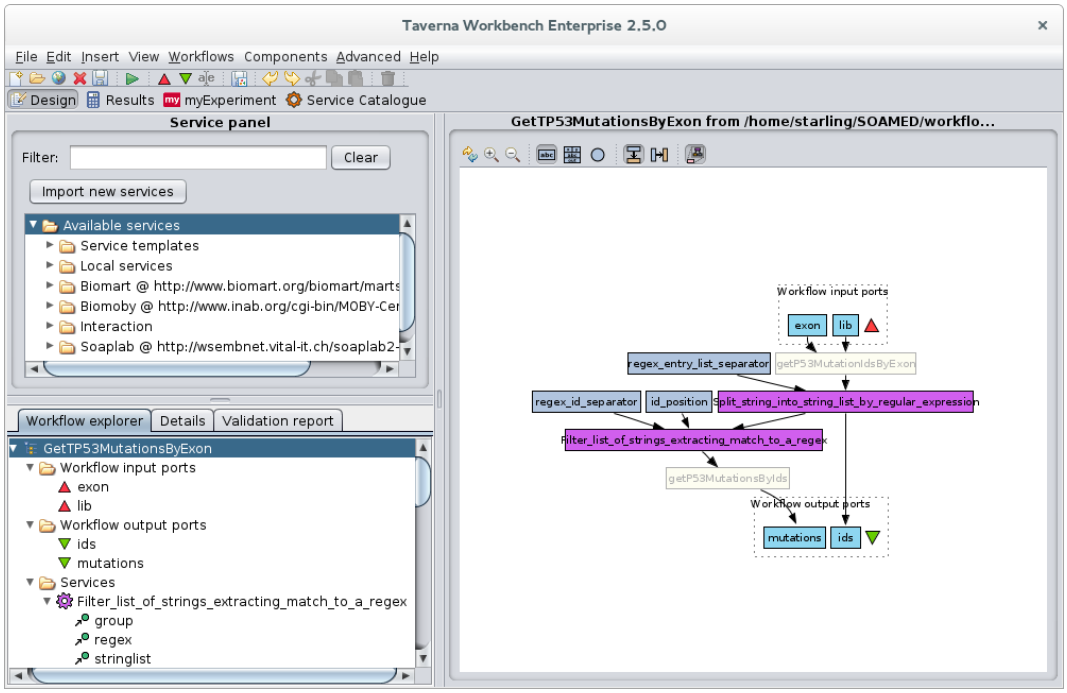


Figure 2.4: SWFM user interface example: Taverna workbench with workflow opened for editing.

as such due to, for instance, the high volume of data to be processed (e.g., when mapping the DNA sequences from a NGS experiment to a reference genome). How unlike the representation of modules is in different SWFM has been reviewed in Section 2.1.1.

Workflow Verification. Comparing Figures 2.3 and 2.4, the modules invoking web-services are grayed out in the workbench view of the workflow, indicating that the services are currently not reachable and the workflow can not be executed properly. Such verification is not only done on complete workflows, but may also be used during authoring of the workflow already.

Workflow Execution, Monitoring, and Recovery. Once a workflow is created, it can usually be executed directly in or from within the SWFM itself. Where actual execution takes place - locally or on a cluster - depends on the SWFM. Some SWFM provide monitoring capabilities, showing the user which modules of a workflow are running at a given point in time, how many parallel instances of each module exist, and where errors occur. In cases of errors, SWFM often allow recovery of workflow exection on the data collected prior to the error, without the need of recomputing intermediate results. This is especially useful for workflows executed on large data sets.

Provenance Tracking. As the data input by the user passes through each module, it is modified, enriched, or transformed. The record of how the data was subsequently changed by the workflow's modules is referred to as provenance, which some SWFM allow to record. Using provenance, the user is able to trace each piece of output data back to the input data it originates from, observing exactly how the output was derived.

To what extent each of these features is supported, depends on the concrete SWFM. And, despite the comfort these features offer to workflow authors, creating a scientific workflow is still a laborious task, that requires deep knowledge of the data processing steps and tools involved in the computational experiment to be run. The data one tool produces often has to be transformed for consumption by the successive one - a task for which sometimes predefined modules from the library provided by the SWFM may be available (as in the example workflow of Fig. 2.4), other times scripts still have to be written for. As such, while much more accessible to domain scientists not trained in programming than traditional, pure scripting approaches, mastering scientific workflow creation is still challenging. For such users it is much more straightforward to re-use existing workflows found in scientific workflow repositories.

2.2.2 Scientific Workflow Repositories

Scientific workflow repositories allow upload of workflows created in an SWFM to store and share them with others. Once published in a repository, these workflows can then be searched, downloaded and executed by other scientists. This not only facilitates reproducibility of the underlying in-silico experiment, but also allows easy re-use of (usually) proven computational methods, or repurposing of a workflow for a different, but similar experiment by applying slight modifications.

Next to the uploaded workflow definition itself, scientific workflows in online repositories may have several types of user-provided annotations associated with them. Which types of annotations are supported greatly varies by repository: While, for instance, myExperiment [74] provides user ratings, download statistics, attributions to workflow co-authors, and features of a social (scientific) network, many other repositories provide far less options for workflow annotation. The most common types of annotations are exemplified in Figure 2.5:

- A title ideally summarizes the workflow's function.
- A description gives a more or less detailed explanation of the workflow's functionality as free-form text.
- Keyword tags assign categories of specific functionality or areas of application to the workflow.

To what extent these annotations are available strongly depends on the concrete repository. Table 2.1 gives an overview of some popular public scientific workflow

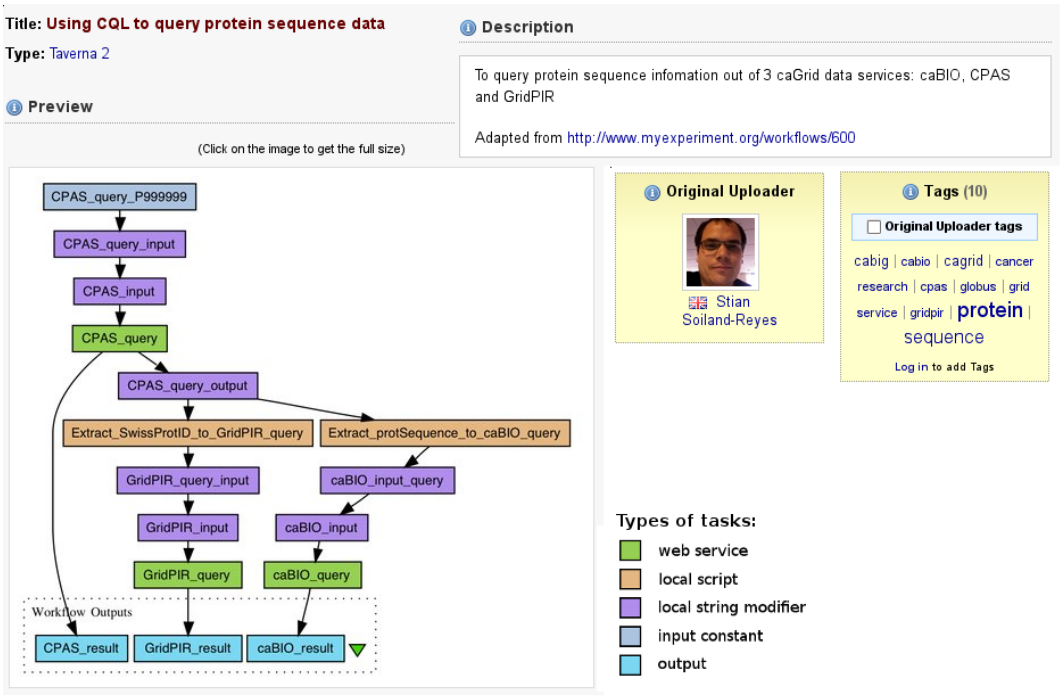


Figure 2.5: Example of a scientific workflow and its annotations from an online repository. Adapted from http://www.myexperiment.org/workflows/752.

repositories and repositories associated with popular SWFM, listing their size in terms of the number of workflows they contain and their respective workflow formats - and (roughly) indicating for how many workflows each type of annotation is available. For some repositories annotations are well available. Other repositories, while offering the functionality to add annotations, hardly contain any. Currently, this greatly impedes their potential for supporting users in discovery of the workflows they contain.

Scientific Workflow Discovery

Scientific workflow discovery is the process of finding that or those workflows in an online repository that match a user's data processing needs. Providing adequate means for such discovery to a repository's users is of vital importance for scientific workflows to reach their target audience, the scientist [21]. Comprehensive support for the discovery of scientific workflows has also been identified as a bottleneck for their re-use in [44].

The observations on the availability of annotations in existing online repositories presented in Table 2.1 are all the more interesting as the prevalent means currently offered by repositories for discovering the workflows they store - apart from browsing lists of hundreds of workflows - critically depends on such annotations: keyword

Repository	App. no. of Workflows	Workflow Format	Annotations		
			title	description	tags
CrowdLabs [64]	200	VisTrails	+	-	-
Galaxy [45] public repository	225	Galaxy	+	~	-
Kepler [14] component library	25	Kepler	+	~	n.a.
myExperiment [74]	2600	any, mostly Taverna	+	+	+
SHIWA [3]	250	any, mostly WS-PGRADE & Taverna	+	+	~

Table 2.1: Overview of sizes and workflow formats of public scientific workflow repositories, together with annotation frequency (+: always/often available , ~: balanced availability, -: seldom available, n.a: option not provided; data as of Nov 2014).

search. The definition, modules, and graph structure of the scientific workflows themselves are typically not exploited to further facilitate access to a repository's contents, or to manage the workflows they contain, even while scientists themselves have identified the structure of a scientific workflow to play an important role in discovering and selecting appropriate workflows [43].

This greatly limits discoverability of scientific workflows. In keyword search, a user enters a number of keywords which are then matched against the annotations stored with the workflows to retrieve suitable ones. Of course, this only works if workflows have annotations, and if the user's keywords match the keywords used by the workflow author to describe the workflow's functionality. Often enough, only few workflows are returned, possibly missing the most suitable workflows due to a mismatch of the keywords used. Or, when the used keywords are too generic and the repository is large enough, long lists of workflows are presented to the user - again leaving him or her to manual inspection of each workflow returned by browsing through the list.

Clearly, this current state of scientific workflow discovery needs to be improved for users to make best use of scientific workflows, their repositories, and the promises they convey for the exchange of scientific knowledge. For instance, the status of keyword search could greatly be improved, if, for the case of using wrong keywords, those workflows matching the query would be used to automatically find other workflows in the repository, which provide similar functionality, but don't contain the respective keywords. In the case of long lists of workflows resulting from a search, workflows in the list providing equivalent or closely related functionality could be grouped to reduce the number of workflows (or groups thereof) the user would need to inspect, or even to suggest refinements of the initial query.

Other possible improvements could include (possibly hierarchical) clustering of scientific workflows by their functionality to provide a catalog for easy browsing; the suggestion of workflows providing similar analyses when viewing a specific workflow in a repository; or even the suggestion of such similar workflows in the scientific workflow management system itself, based on the workflow a user is executing, authoring or merely planning. Any such improvements, of course, first require a method to identify whether two given workflows are similar at all.

2.3 Similarity of (Scientific) Workflows

2.3.1 Functional Similarity of Scientific Workflows

Building on the fundamental properties of scientific workflows described in the previous section, and the properties of the repositories these workflows are stored in, this thesis targets the investigation of scientific workflow comparison with respect to their *functional similarity*. That is, we are interested in automated ways of measuring whether the overall functionality offered by two workflows is similar or not, and, ideally, to what degree it is similar. Capturing such functional similarity is not quite straighforward, even for human judgement. Are two workflows functionally similar only if they accept the same types of input data, but provide different, yet, again, similar analyses over them? Or may two workflows be regarded as functionally similar if the analyses they provide are indeed similar, but operate on different types of input data? Do similar outputs indicate similar functionality? Or do they only do so when the workflows' inputs are similar as well?

Whether two workflows are functionally similar will depend on the domain of workflow application, the user searching for functionally similar workflows, and the specific problem a given workflow addresses. For algorithmic accessibility of the problem of defining functional similarity of scientific workflows, we here focus on their *measurable* functionality - using the information provided by workflows in online repositories. Whether algorithms relying on this measurable information provide satisfying results will have to be subjected to evaluation by human judgement. In the following, we give a overview of existing approaches trying to measure the functional similarity of scientific workflows. While evaluations provided with these measures range from manual inspection of a methods output (e.g., [25]) to comparison against a baseline provided by another method (e.g., [39]), no human curated gold standard corpus of sufficient size is available for comprehensive evaluation of different methods. For the purposes of this thesis, acquiring such a corpus will be indispensible.

2.3.2 Approaches to Workflow Similarity

As explicated in Section 2.1, scientific workflows are typically modeled as dataflows. These contain global input and output ports, modules which operate on the data, and datalinks which connect the former three amongst each other and define the flow

of data from one module to the next along the processing pipeline. Each module has attributes associated with it, such as a label the workflow author has assigned to it, the type of operation to be carried out on the data, and, if applicable, a set of static, data independent parameters. Upon upload to a repository, scientific workflows may receive a title, a description, or keyword tags, and are associated with their uploading author.

These annotations, along with the graph structure of scientific workflows itself, can be exploited to compare scientific workflows and establish a similarity measure for them. Depending on which properties of workflows they use, existing approaches to assessing scientific workflow similarity can be classified as being either annotation-based only or based on workflow structure. In the following, we briefly survey existing approaches along this classification, and discuss options for approaches that use other kinds of data. In addition to approaches targetted at scientific workflows, we also look at work in the related fields of business process models. While there exist differences between the two models [90, 61], methods established for the latter offer inspiration or might even be directly applicable to scientific workflows as well.

Annotation-based Approaches

The textual attributes of scientific workflows can be exploited for clustering scientific workflows. The authors of [25] applied text mining methods (mutual terms in descriptions) to cluster scientific workflows based on their descriptions. The authors argue that using solely the textual descriptions allows workflow comparison across different workflow environments, including different SWFM, where the different workflow formats limit the potential for structure based comparison. Yet, the study fails to provide a comprehensive evaluation of the used method.

[82] explores the use of tags assigned to workflows for workflow clustering and compares the results to the use of features derived from the modules present in workflows. Their application scenario targets the use of workflow clustering into categories for easier browsing of scientific workflow repositories. They conclude that both tag and modules-feature based clustering methods perform better on repositories with a fairly limited range of tasks. For repositories with workflows targeting a broader range of tasks, and using a more diverse set of modules, such as myExperiment, they find the tag based clustering approach superior to the module-feature based method.

A general drawback to annotation based methods is, of course, that they have to rely on such annotations to be available. As we have seen in Section 2.2.2, often enough this is not the case. Furthermore, using annotations for assessment of workflow similarity, ultimately depends on users to use common terms (or even languages) to describe and tag workflow functionality. For workflows where this is not the case, or annotations are not available, other sources of information have to be resorted to.

Structure-based Approaches

The functionality of a workflow is determined by which type of input data it accepts, which modules operate on the data in which order, and what type of result it produces. Following this enumeration, two workflows may be functionally similar if they accept similar types of inputs, use similar modules to operate on the data, or produce similar types of outputs. As described in Section 2.1.1, workflows created by current SWFM most often don't use semantic typing of inputs or outputs - greatly limiting their use to the assessment of functional similarity. Of course, the same applies to the workflows' modules themselves, for which mostly technical and syntactic attributes are available. As such, previous work on assessing scientific workflow similarity from their structure has focussed on distilling such similarity from the topology of the workflows' modules. A number of different approaches have been tried for various use cases of workflow discovery:

In [43], subgraph isomorphism is used for scientific workflow matching. A query workflow is matched against a repository of workflows to find the most similar ones. The authors exploit the graph matcher presented in [66] which is originally targetted at matching a large query graph against a repository of smaller graphs. While this original intention is not followed in [43], the authors note that the use of common subgraphs for workflow indexing, and subsequent matching against the index rather than against the workflows themselves might yield further improvements in the matching process, especially regarding performance. Subgraph-isomorphism has also been investigated to compare BPEL workflows [24] using error correcting subgraph-isomorphism based on graph edit operations.

Graph kernels are used in [39]. The authors first extract frequent subgraphs from a set of scientific workflows and use these as features to form graph kernels. These kernels are used to classify workflows and a) make workflow recommendations given an input workflow, b) derive workflow tags from workflows containing similar substructures, and c) extract meaningful common patterns from a set of workflows. The results are evaluated against workflow clusters derived from the tags assigned to the workflows on myExperiment. It is found that a metadata based approach based on workflow descriptions, tags, and authors outperforms the proposed graph matching method.

The authors of [88] use the SUBDUE package [66] to compute scientific workflow similarity using graph-edit-distance. An evaluation of matching results is not provided. [78] presents a method for clustering of workflows based on cumulative similarity values of the module sets they are comprised of, not considering the graph nature of the workflow as a whole. [76] investigates the use of maximum common isomorphic subgraphs in comparison to module label vector similarity, i.e. the collection of labels of the modules used in a workflow. They find that for the set of workflows studied, both methods deliver similar results in the task of workflow clustering.

All structural approaches for workflow similarity greatly depend on matchings between workflow modules. Mapping modules of different scientific workflows onto

each other is not trivial, and will need to be a core issue of scientific workflow similarity. Yet, of the aforementioned publications, none but [78] fully describes the process of node identification, i.e., matching one workflow's modules to the modules of another workflow. From the figures provided in [39] we conclude that node identification is based solely on the type of module the node represents, e.g., *local*, *script*, or *web-service*. For the others, it seems only module labels were used.

Similarity of business process models has been studied much more in detail, so far, when compared to scientific workflows. [34] surveys several matching approaches for BPM similarity. In [32], the authors directly compare three different methods of computing similarity. The first exploits only the labels of workflow elements. The second uses the labels and the structure of the model by computing the graph-edit-distance between the labeled graph representations of the models. The third computes the behavioural similarity derived from the causal relationships between elements of two business process models. The experimental evaluation shows that the structural similarity metric slightly outperforms the other two methods. This work is continued in [33] where the authors explore different graph matching algorithms for BPM comparison.

Abstracting from scientific workflows, more generally, much research exists on graph matching. While scientific workflows typically do not show all possible graph constellations, but are limited to a subset of those [13], general graph matching algorithms which have not yet been exploited for the concrete scenario of scientific workflow comparison, may well be applicable. [4] and [22] give comprehensive overviews of different graph matching algorithms.

Other Approaches

So far, we have looked at approaches for comparing scientific workflows which use the properties of the workflows themselves or annotations available from some current scientific workflow repositories. Additionally, external sources of information can be used to augment the amount of available metadata. For instance [42] proposes an approach to use annotations derived from external component catalogs to augment the workflow descriptions. While the concrete method relies on a well annotated component catalog which is, as such, not available for arbitrary components of scientific workflows, the general idea of using external information about workflow elements is interesting.

A similarity flooding strategy is used in [77] to determine workflow similarity based on module correspondences between VisTrails workflows, where the *semantic* information of a module's operation underlying the module's mapping is well defined - which is most typically not the case for scientific workflows found in online repositories. Similarly, [8] use manually added semantic annotations on modules and datalinks to compare workflows.

Another possibility for comparing scientific workflows is the use of provenance

information recorded from their execution. Research on information discovery from provenance traces has recently received much attention [26], as it enables the user of the workflow to understand how data is processed and modified by the workflow during execution. [6] explores the problem of comparing of different provenance traces from the same workflow. Using such traces from different workflows might be a very powerful method to determine whether two modules share the same or similar functionality - not relying on their technical attributes or annotations, but deriving information from their behaviour on observed input and output data. Next to the difficulties entailed in such data-wise comparison itself, a current obstacle is that provenance traces are generally not available. Specialised repositories for storing provenance traces are just starting to emerge [2, 49, 67]. In this thesis, we focus on similarity measures for scientific workflows using the data available in current repositories.

3 Component (Re)Use in Public Scientific Workflow Repositories

Similarity measures between scientific workflows, first require a method for identifying shared elements, e.g., shared analysis tools [84]. Only then we can identify similar workflows in a repository, making similarity search worth while. Therefore, an important step towards the investigation of workflow similarity searching is the analysis of the workflows contained in a repository, in particular to find elements that are shared across workflows. For web-services such an analysis is presented in [83], who found that only few web-services are used in more than one workflow. Yet, another analysis of modules contained in the workflows stored in myExperiment [87], revealed that the majority of basic tasks are locally executed modules and not web services. In [82], functional properties of several types of modules, including local ones, are used as features to classify workflows. Yet, no differentiation is made between single types of modules after extraction, hindering fine-grained analysis of shared elements. Module labels are used for identification in [43] in order to match a very limited set of workflows using subgraph-isomorphism. From our experience, this label-based approach is not generally applicable to large public collections of scientific workflows: The broad author base results in substantially heterogeneous labels.

We believe that all the above studies failed to give a comprehensive account on the degree of sharing between workflows in a repository. In particular, three key aspects were not adequately considered.

1. To determine elements shared by different workflows, one needs to establish a method to test the identity of (or similarity between) two elements from different workflows. While the identity of web-services is determined by service name and operation, this approach is not generalizable to arbitrary modules. In [82], the authors build such a method by extracting features from modules, yet omit important information; for instance, local modules are only identified by their respective class, disregarding their actual instantiation parameters.

2. From a structural perspective, workflows can be described at three levels of functionality: single modules (typically treated as black boxes), groups of modules organized as dataflows, and whole workflows. As defined in Section 2.1.1, a workflow always consists of one top-level dataflow which contains modules

and potentially includes subworkflows, that is, further, nested dataflows. However, previous work only considered sharing at the level of modules, although sharing of more coarse-grained functional units such as dataflows is equally important.

3. Another important aspect of studying workflow connectivity is authorship. We argue that true sharing of workflow elements is only achieved if elements (modules or dataflows) are shared between workflows authored by different persons; however, authorship has, to the best of our knowledge, not been considered in any of the prior repository analyses. Mere statistics for web-service usage across workflows, for instance, leave the question unanswered, whether preferences in service usage vary between authors, or which services are used by the broadest author base.

This chapter investigates and presents results on all of these three issues. We describe the results of a comprehensive study on reuse in the largest open scientific workflow repository to-date, myExperiment.org. We discuss and evaluate different methods to establish element identity at all three levels of workflows, i.e., at the modules level, the dataflow level, and the workflow level, and further refine the results based on classes of modules. Overall, we find that elements are shared at all levels, but that most sharing only affects "trivial" elements, e.g., modules for providing input parameters or type conversions. Furthermore, we provide a detailed analysis of cross-author reuse and show that much of the previously identified reuse is by single authors. Still, a significant number of non-trivial elements are shared cross-author, and we present first results on how these can be used to identify clusters of related workflows.

The remainder of this chapter is structured as follows. We first describe in Section 3.1 the data sets and the methods used for identifying workflow elements. In Section 3.2 we present the results of our analysis on each of the aforementioned levels. Section 3.3 discusses our findings, and provides an outlook on future research.

This work appeared in [81].

3.1 Materials and Methods

3.1.1 Data Sets

We study the reuse of elements in myExperiment.org, which, to our knowledge, is the public scientific workflow repository containing the highest number of publicly available workflows (see Table 2.1). myExperiment allows upload of workflows from several systems, but approximately 85% of its content are workflows for Taverna [68]. Taverna workflows can appear in two different formats, namely *scufl* (used by Taverna 1) and *t2flow* (used by Taverna 2). In this work, we only consider Taverna workflows, yet, in either format.

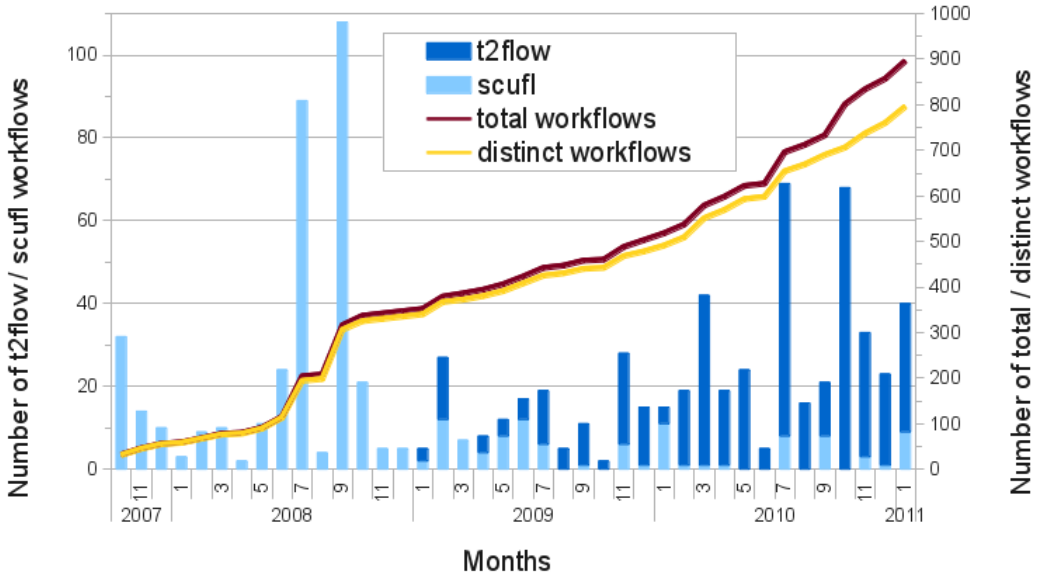


Figure 3.1: Taverna workflows uploaded to myExperiment by month (left hand scale), and the numbers of overall and distinct workflows in the myExperiment repository the uploads amount to (right hand scale).

For these two types, Figure 3.1 shows the number of workflows submitted to myExperiment per month. The first Taverna 2 workflows appeared in January 2009 when the new version of Taverna was released. But, even in recent times, still a sizable number of Taverna 1 workflows are being uploaded. Note that Taverna 2 can load and process both *scufl* and *t2flow*. Figure 3.1 also shows a steady overall growth in total available workflows. This growth of the repository is accompanied by an increase in duplicate workflows. Most but not all of these apparent redundancies are caused by the format change (the same workflow uploaded once as a *scufl* and once as a *t2flow* workflow). We get back to this point in section 3.2.3.

498 workflows in *scufl* format, and 449 in *t2flow* format were downloaded from myExperiment.org, which, at the time of download in December 2011, were all Taverna workflows available in this repository. Of the 498 *scufl* files, 449 could be converted to *t2flow* by manually loading them into the Taverna 2 Workbench and storing them in the new format. 49 files showed format inconsistencies and were removed from further analysis. Altogether, our analysis is comprised of 898 workflows.

Objects of study

Figure 3.2 provides an example workflow from our study set, illustrating the three functional levels we are studying. This workflow has two global inputs and four global outputs; it is composed of one top-level dataflow and one nested dataflow

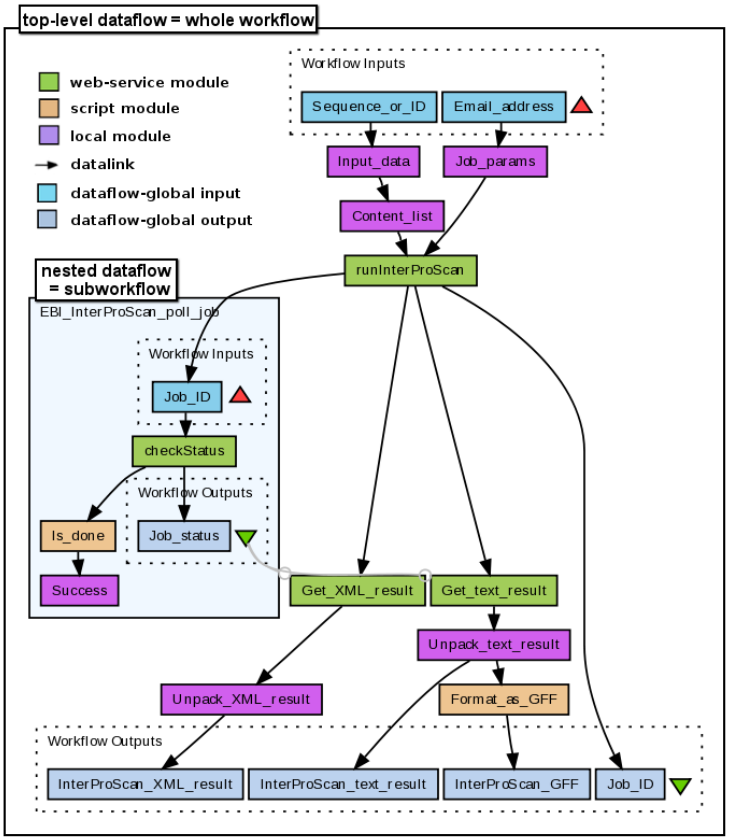


Figure 3.2: Example of workflow (myExperiment id 204)

named *EBI_InterProScan_poll_job*. In accordance with our definition of scientific workflows in Section 2.1.1, a Taverna workflow is generally defined by one top-level dataflow and may contain nested dataflows. Each dataflow has its own main inputs and outputs and consists of one or more modules representing activities operating on data. Modules are connected by datalinks, describing the flow of data from one module's output(s) to the next module's input(s). Each module has a type and a configuration, where the type denotes the executable class and the configuration contains the parameters passed to the class for instantiation. For example, the configuration of a *WSDLActivity* module denotes the url of the WSDL document to enact and the operation to call with each piece of data received by the module via its input ports. The specifics of Taverna's workflow model are presented in more detail in [68].

The 898 workflows used in our analysis contain a total of 1,431 dataflows, including both 898 top-level and 533 nested dataflows, and a total of 10,242 modules. On average, each workflow uses 1.6 dataflows and 11.4 modules, where the largest workflow has 455 modules and the smallest has 1 module. In terms of dataflows,

the largest workflow contains 19 nested dataflows.

In myExperiment, each workflow is further accompanied by metadata provided in RDF-format including title, description, author and version information, and creation and modification dates. For each workflow in our set, we downloaded its accompanying metadata and specifically extracted the workflows author.

3.1.2 Identifying Shared Workflow Elements

Investigating interconnections between (parts of) workflows and cross-author reuse necessitates precise methods for identifying elements. At first glance, elements can be deemed identical if they are functionally equivalent, i.e, if they, for every possible input, always produce the same output. However, such a stringent definition is not helpful as it is fundamentally undecidable if two programs are identical in this sense. Also, this definition of identity is not advisable for our setting because we ultimately work in a retrieval setting: Imagine a user has chosen a workflow X for analyzing her data set and is interested in exploring workflows that perform similar analyses to learn about alternatives; such a user is naturally interested in elements (and workflows) implementing a similar method, not an identical one [21]. In the following, we discuss various methods to account for this fuzzyness in comparing workflow elements at each of the three levels, i.e., at module level, at dataflow level, and at workflow level.

Identifying modules

The configuration of each module specified in the *t2flow* workflow definition file contains both the type of module and the code it is to be instantiated with. After having cleansed from whitespace all the configurations, we used them as the way to identify identical modules. We thus assume modules to be identical if their cleansed configurations match exactly. Clearly, this is a very strict definition of identity: however, please note that it still leads to a meaningful definition of workflow similarity as workflows typically contain multiple (identical or not) modules. We shall explore the impact of relaxing this definition in Section 3.3.

Identifying dataflows

Both myExperiment and Taverna provide identifiers for dataflows: while myExperiment assigns an integer to each workflow uploaded, Taverna assigns a 36 character, alphanumeric string value to each dataflow created. When a dataflow is included in a workflow as a subworkflow, the Taverna id is used to reference the dataflow. The way this id is determined is not defined by Taverna; although it seems to be meant to be able to uniquely identify a dataflow across systems (like a checksum), we found this id to be not sufficient for this task. We both found the same Taverna id to be assigned to different dataflows (e.g., workflows 1702 and 1703) as well as identical dataflows having different Taverna ids (e.g., workflows 359 and 506). Fur-

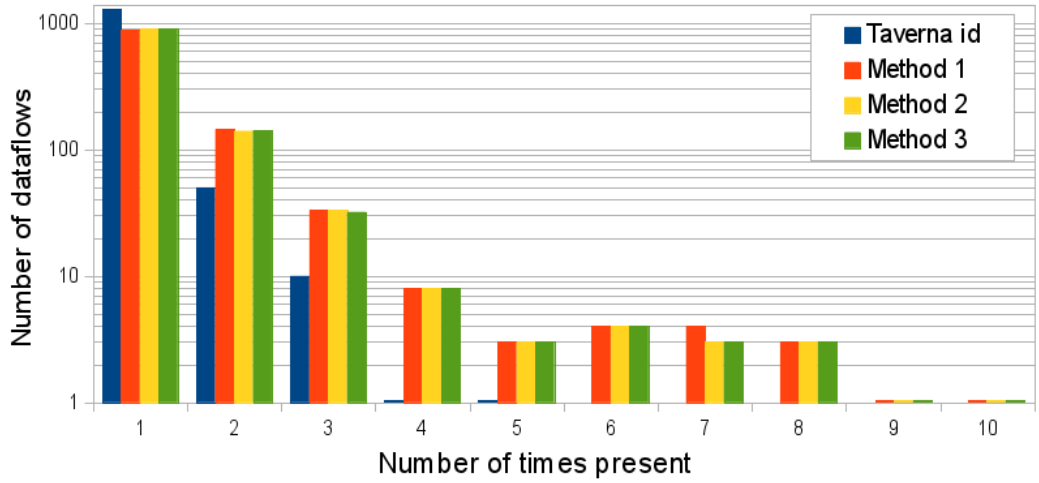


Figure 3.3: Occurrences of distinct dataflows in the analysis set for several methods of identification

thermore, using an id would restrict our analysis to identity of dataflows. Therefore, we devised three alternative ways to compare two dataflows:

- Method 1: We consider two dataflows as shared if the sets of modules are identical;
- Method 2: We consider two dataflows as shared if the sets of modules and the numbers of datalinks are identical;
- Method 3: We consider two dataflows as shared if their sets of modules, the numbers of datalinks, and the numbers of global inputs and global outputs are identical.

Note that we use multisets here: if two modules contained in a dataflow's module set have identical cleansed configurations, they are still both included in the set. Figure 3.3 gives an overview of the dataflow occurrences found using Taverna ids and each of our three methods. A first observation is that the number of dataflows occurring (i.e. used) only once is just above 80% of the total number of distinct dataflows when identification is based on method 1, 2, or 3 while it is 95% when the identification is based on Taverna ids. Assuming that two dataflows must be highly similar already if they consist of the same set of modules, we conclude that any of the three methods appear more suitable than Taverna ids to study reuse across workflows. Second, while method 1 assimilates several dataflows which do not share a Taverna id, the addition of datalinks by method 2 - not surprisingly - re-adds some distinction. However, the fact that the total number of distinct dataflows only increases slightly (from 1,071 to 1,081) when going from method 1 to method 2

shows that almost every identical set of modules is connected by the same number of datalinks.

To gain more insight into this finding, we looked more closely into the dataflows which had the same set of modules but were connected by different numbers of links. We found that those only differ in the way they deal with outputs (use of additional module outputs to propagate results to the final output). We further studied the difference in results between methods 2 and 3 which affect only a group of three dataflows (contained in workflows 1214, 1512 and 1513) which method 3 splits into two groups of one and two dataflows, respectively. However, the differences in these dataflows, again, only affects the number of inputs and outputs, but not the analysis performed by the workflow itself.

Based on these observations, we decided to continue our study using only method 1. In the following, when we speak of 'distinct' dataflows, we thus mean them to differ by their respective sets of modules.

Identifying workflows

We follow the same thoughts for comparing workflows as for comparing dataflows. We consider two workflows as identical if they are build from the exact same set of dataflows and modules.

Note that we did not include the textual descriptions of modules and dataflows in the process of identification. We did find workflows that are deemed related when using the identification scheme established above but differ in the names assigned to some of their modules. This difference does, however, not cause any structural or functional divergence.

3.2 Results

Using method 1 outlined before, we found 3,598 of 10,242 modules, 1,071 of 1,431 dataflows and 792 of 898 workflows to be distinct. In the following, we look at workflow interconnections and cross-author reuse on each of these structural levels separately and also break up our analysis by different module categories.

3.2.1 Modules

We first looked at the general usage of modules by comparing the total numbers of module occurrences with the number of distinct modules, and their use across workflows and authors. On average, each module is used 2.85 times in 2.24 workflows by 1.31 authors. Figure 3.4 shows the relative usage frequencies for all modules in our set. Overall reuse of modules and cross-workflow reuse of modules closely correlate (Pearson Correlation Coefficient > 0.99, p-value $\ll 0.01$). The slight difference is caused by single modules being used more than once in single workflows. Cross-author reuse, on the other hand, is much lower, indicating that workflow authors reuse their own workflow components more often than those created by others.

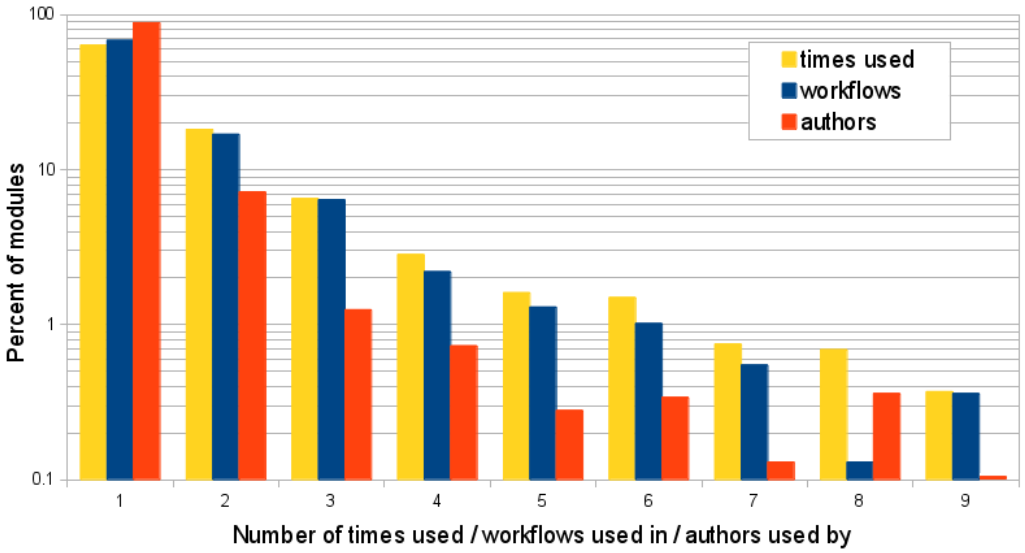


Figure 3.4: Usage of modules overall, in workflows and by authors.

Maxima[1] of 314 usages, 177 workflows and 39 authors and minima of 1 for single module usage call for a more detailed investigation. To this end, we used the categorization established in [87], organizing Taverna modules into the four main categories *local, script, subworkflow*, and *web-service*. Note that subworkflows are nested dataflows which will be discussed in more detail in Section 3.2.2. Within these categories, modules are further divided into subcategories based on functional or technological characteristics. The authors of [87] showed that overall usage of modules varies greatly between these categories and subcategories. Here, we extend their analysis by considering the reuse frequencies of modules within each category.

Usage statistics by category and by subcategory are shown in Table 3.1, listing total and distinct numbers of modules for each subcategory, together with the numbers of workflows and authors using modules from these subcategories. Figure 3.5 exemplifies total, workflow-based, and author-based relative usage distributions in subcategories *beanshell* and *data conversion*.

For the ***local*** category 27% of modules are distinct with a reuse rate of 44%. Modules from this category access functionality provided and executed by the SWFM. This category is particularly interesting for workflow interconnections, as all of its subcategories show comparably high reuse rates, overall as well as across workflows and authors. This is especially true for *conditional, user interaction, operation, database access, testing* and *util*: While for most module subcategories usage distributions are similar to those shown in Figure 3.5, the former exhibit a much broader distribution. The reason for this fact is that many of the respective modules are from

[1] Usage maxima not shown in Figure 3.4 for better visualization

(Sub)category	Modules total	Modules distinct	Modules reused	Times used avg	Times used stddev	Times used max	Workflows avg	Workflows stddev	Workflows max	Authors avg	Authors stddev	Authors max
local	6518	1786	777	3.65	13.91	314	2.71	7.30	177	1.47	2.16	39
cdk	63	56	7	1.12	0.33	2	1.08	0.28	2	1.00	0.00	1
conditional	109	2	2	54.50	19.50	74	38.50	13.50	52	10.50	3.50	14
string constant	2817	844	387	3.33	9.83	173	2.71	6.21	114	1.44	1.81	28
data conversion	2979	771	307	3.86	17.84	314	2.56	8.27	177	1.43	2.34	39
user interaction	62	9	8	6.88	6.40	22	5.55	4.59	16	2.22	1.54	6
operation	5	2	2	2.50	0.50	3	2.50	0.50	3	1.50	0.50	2
database access	81	13	10	6.23	7.94	28	4.69	5.01	20	2.92	4.21	16
testing	13	6	3	2.16	1.21	4	2.16	1.21	4	1.33	0.47	2
util	389	83	51	4.68	11.25	84	3.73	8.49	63	1.79	3.16	25
script	1393	753	227	1.85	3.69	83	1.60	2.03	30	1.07	0.46	8
beanshell	1333	701	220	1.90	3.81	83	1.63	2.09	30	1.07	0.47	8
r	60	52	7	1.15	0.41	3	1.03	0.19	2	1.01	0.13	2
subworkflow	533	358	91	1.48	1.23	11	1.38	1.01	9	1.03	0.19	2
subwf only	380	279	54	1.36	0.95	9	1.27	0.85	9	1.03	0.17	2
subwf + top	153	79	32	1.93	1.82	11	1.75	1.38	7	1.05	0.21	2
web-service	1798	701	309	2.56	4.27	47	2.14	3.32	36	1.29	1.28	21
biomart	79	49	12	1.61	1.49	7	1.61	1.49	7	1.16	0.50	3
biomoby	58	44	9	1.31	0.66	3	1.31	0.66	3	1.11	0.31	2
cagrid	9	8	1	1.12	0.33	2	1.12	0.33	2	1.00	0.00	1
rest	6	6	0	1.00	0.00	1	1.00	0.00	1	1.00	0.00	1
sadi	5	5	0	1.00	0.00	1	1.00	0.00	1	1.00	0.00	1
soaplab	332	81	46	4.09	5.50	31	3.41	4.69	22	1.53	1.34	8
wsdl	1304	503	241	2.59	4.41	47	2.11	3.33	36	1.29	1.39	21
xmpp	5	5	0	1.00	0.00	1	1.00	0.00	1	1.00	0.00	1

Table 3.1: Statistics on module usage by subcategory.

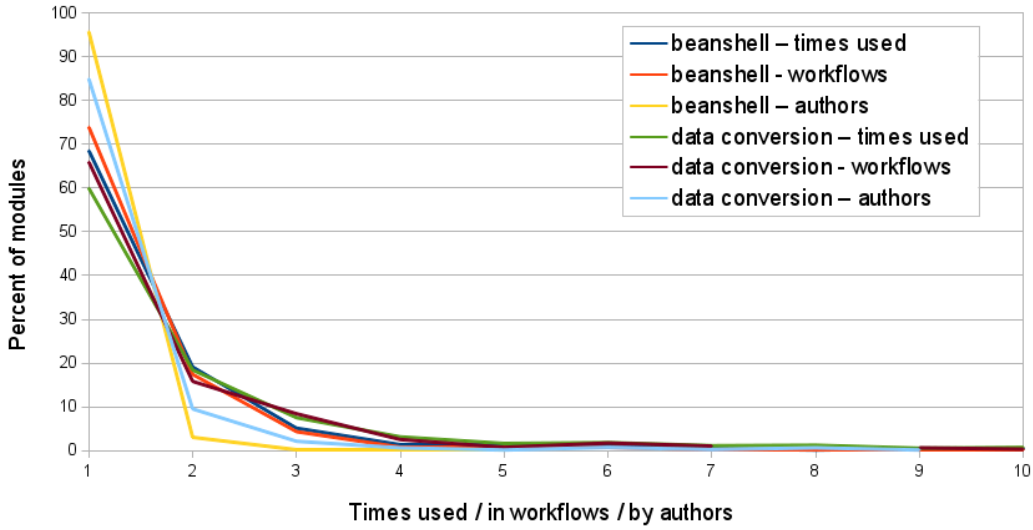


Figure 3.5: Exemplary usage distributions for beanshell and data conversion modules showing relative overall, cross-workflow and cross-author usage frequencies.

the standard set that Taverna provides for building workflows, and used without further modification by the author. Table 3.2, listing the top five most frequently used modules, underpins this finding: These modules provide very common functionality which is likely to be widely used.

54% of all ***script*** modules are distinct, of which 30% are reused. Within this categroy, *R* scripts are far less often used than *beanshell* scripts and hardly used more than once. *Beanshell* scripts, on the other hand, are the third most popular type of modules with only 53% of its instances being distinct. 31% of them are reused. This seems remarkable, as we would expect these modules to contain user-created functionality for data processing. Yet, looking at the author-based distribution of these modules reveals that almost 96% of them are used only by single authors. It appears that users of beanshell modules have personal libraries of such custom-made tools which they reuse quite frequently, while usage of others' tools is rare.

Web-service modules show 39% distinctiveness, and 44% reuse. By far the most popular types of web-service invocations are *soaplab* and *wsdl* modules. 24% of *soaploab* modules and 39% of *wsdl* modules are distinct, and reuse is at 57% and 48%, respectively. As for scripts, reuse across authors is low, with single author usage rates of 78% and 87%, respectively. This gap between overall and cross-author reuse shows quite clearly that authors use and reuse certain web-services preferentially, while these preferences are not too widely shared between workflow authors. An exception to this are some rather popular, well-known web-services, such as *Blasts*

Category	Subcategory	Description	T	W	A
local	data conversion	Regular expression splitter taking an optional regex as input (default ',') and splitting an input string into a list of strings.	314	177	39
local	data conversion	String list merger taking an optional string separator (default *newline character*) and merging an input list of strings into one new string with the original strings separated by the given separator	309	87	25
local	string constant	string constant *newline character*	173	114	28
local	string constant	string constant '1'	157	82	22
local	data conversion	string concatenator for two input strings	118	66	28

Table 3.2: Top 5 most used modules (T: times used; W: workflows used in; A: authors used by).

SimpleSearch.

Returning from the categorized to the global view, Figure 3.6 shows the 300 most frequently used modules and their cross-workflow and cross-author reuse. Overall usage counts clearly follow a Zipf-like distribution. Zipf's law [93] states that when ranking words in some corpus of natural language by their frequency, the rank of any word in the resulting frequency table is inversely proportional to its frequency. Carrying over this distribution to modules in scientific workflows, it means that only few modules are used very often, while usage of the vast majority of modules is very sparse. The corresponding counts for reuse across workflows and authors exhibit the same trend. Yet, they show peaks of increased reuse which mostly are synchronized between the two. These peaks are caused by the aforementioned Taverna built-in modules.

3.2.2 Dataflows

As shown in Figure 3.7, the pattern of usage for dataflows closely follows that of single modules. In contrast, overall reuse is lower by 20%: Over 80% of dataflows are used only once, and only 5% used more than twice. Usage across workflows is slightly lower, implying that some workflows use single dataflows multiple times. 1,038 dataflows are used by single authors, 29 by two authors, and 1 each by 3,4,6 and 7 authors, resulting in an overall of only 3% cross-author reuse for dataflows.

As described in section 3.1.1, dataflows can be either top-level or nested. A top-level dataflow is the same as the entire workflow, while a nested dataflow is a subworkflow. Due to this dual nature of dataflows, three different cases of reuse may occur: (a) Reuse of whole workflows as whole workflows; (b) Reuse of subworkflows as subworkflows; (c) Reuse of whole workflows as subworkflows and vice versa.

Case (a) can be deemed undetectable when looking only at a repository, as reusing a workflow does not mean re-uploading it (the reason why myExperiment still

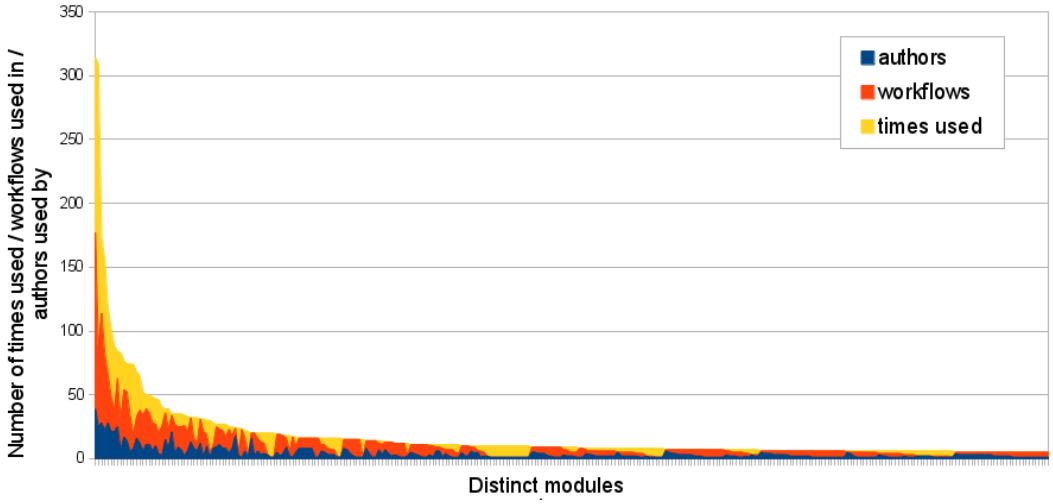


Figure 3.6: Usage counts of the 300 most used modules showing a Zipf-like distribution.

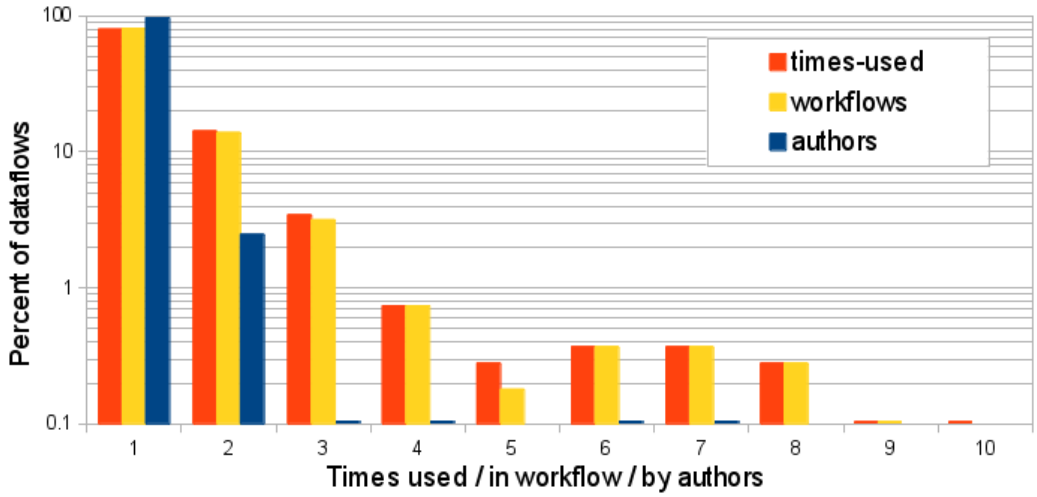


Figure 3.7: Dataflow occurrences in total, in workflows and by authors.

contains duplicates is investigated in the next section). To distinguish cases (b) and (c), we grouped all dataflows in our analysis by their appearance as workflows or subworkflows. 380 (75% distinct) dataflows are only present as nested, but not as top-level dataflows. 153 (55%) nested dataflows are also published as standalone top-level dataflows. For the second group we identified a total of 86 standalone workflows which are used as subworkflows in these 153 cases. Numbers of reuse for the two groups are also shown in Table 3.1.

We did not find significant differences in cross-author reuse between these groups. For cross-workflow and overall reuse, on the other hand, major differences exist: Numbers are as high as 40% for those subworkflows which have a corresponding standalone workflow published[2], while for those that don't they are at only 19%. This can be interpreted in two ways. First, it indicates that authors publish the dataflows they use most often as standalone workflows. This eases their inclusion as nested, functional components inside other workflows. On the other hand, the finding that such dataflows are, for the most part, not used by different authors in derivative work shows that modular extension of existing workflows created by others is still uncommon.

3.2.3 Workflows (Top-Level Dataflows)

Of the 898 workflows, 83 appear more than once in the repository, 19 of which where uploaded by more than one user. This indicates that there are users which upload workflows which are equal (by our definition of identity) to already existing ones. Figure 3.8 shows author contributions of workflows and their dataflows, both total and distinct. It reveals that a single user (the one with the highest overall number of workflows uploaded) is responsible for the majority of the cases of duplicate workflows. By looking into this in more detail, we found that all of this user's duplicates are caused by equivalent workflows being uploaded in both *scufl* and *t2flow* formats. Figure 3.8 also shows that this user alone has authored 23% of all workflows analyzed. Communication with the respective author, who is part of the Taverna development team, revealed that most of his workflows serve the purpose of testing the functionality of Taverna-provided modules and giving examples for their usage. The remainder of duplicate workflows is largely due to users following tutorials including uploads of workflows to myExperiment: They upload an unmodified workflow.

As another interesting finding, the top 10 single authors (groups 14 through 23) have created 554 worfklows, i.e., app. 62% of all workflows in our analysis set. Conversely, 43% of all 124 authors have only created one workflow.

3.3 Discussion

We studied reuse of elements in scientific workflows using the to-date largest public scientific workflow repository. In contrast to previous work, our analysis covers all types of modules and looks into reuse not only at the general level, but also by author and by module category. In this section, we discuss our results and some of the decisions we took when defining our methods. We also point to how our findings should influence next steps towards meaningful similarity measures for scientific workflows.

[2]The standalone workflow itself was not included in reuse computation.

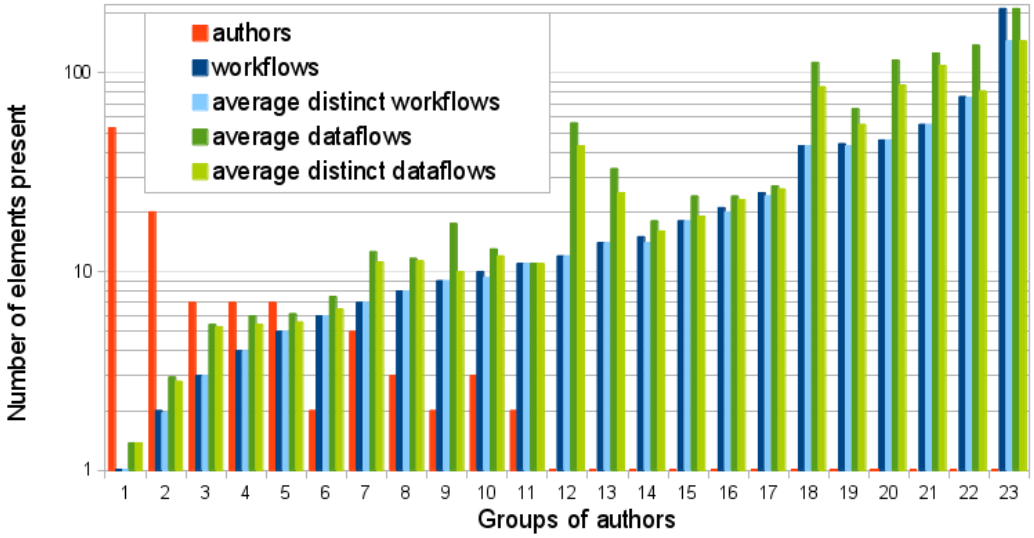


Figure 3.8: Authors grouped by the total number of workflows they have created. Total amounts of workflows, and averages of distinct workflows, and total and distinct dataflows shown for each group.

Module identification. We used exact matching of the modules' configurations to determine module identity. Thus, we only identify verbatim syntactic reuse. To assess the impact of this limitation, we computed pairwise Levenshtein edit distance between all modules[3]. We compared the level of reuse using our strict definition of module identity with one that assumes modules as identical if their edit distance is below a given threshold. Results are shown in Figure 3.9. Clearly, relaxing the threshold even until 20% difference does not have a significant impact on the number of distinct modules. On the other hand, relaxing syntactic similarity comes with the risk of assimilating modules with different functionality, and thus, different usage intent; a problem also present in [82].

Of course, string similarity is a purely syntactic measure, while modules identity in reality is a semantic issue. Thus, more differentiated solutions could be explored. [78] additionally compares the numbers and types of input/output ports, while [42] suggested to use semantic or functional information provided by catalogues of tools. However, both approaches have their problems. Regarding the former, one must consider that the majority of modules in our data set only have a single input and a single output port, mostly of type String. On the other hand, the latter depends on the existence of well-curated ontologies for describing the function of a module, and on authors using these ontologies to tag their modules[4]. Another option would

[3] 15 modules with configurations longer that 20,000 characters were excluded from computation. Results were normalized using the lengths of the compared module configurations. Only modules from equal functional subcategories where compared.

[4] Note that myExperiment allows tagging of workflows, but does not enforce a fixed vocabulary

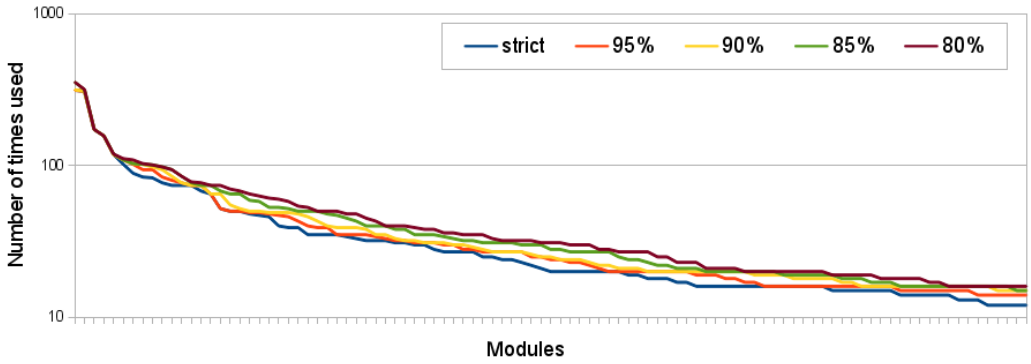


Figure 3.9: Change in overall usage frequencies for modules when matched by 95, 90, 85, and 80 percent similarity regarding their Levenshtein edit distance.

be to exploit provenance traces to infer functional similarity between modules. If two sets of provenance data can be mapped onto each other, so might the modules responsible for the corresponding changes in the data. Unfortunately, repositories specifically targeted at collecting provenance are just starting to emerge [67], and provenance data is currently not available for large groups of workflows.

Reuse characteristics. As we have shown, the most commonly used modules are generic string operations and string constants. More generally, the main glue points of workflows are modules with non-specific functionality, provided by the system and used as-is. As such, these modules provide neither specialized, nor author-created functionality, limiting their usefulness for detecting both cross-workflow and cross-author reuse. Other types of modules are most often only used across workflows created by single authors, and their reuse frequencies are lower than those of unspecific ones. Yet, these modules are especially interesting: The custom-made nature of modules from the *script* category differentiates them from other modules in terms of their highly user-provided functionality. Thus, if such a module is found more than once, it is an indication of non-trivial reuse. The case is similar for web-services. On the other hand, even apparently trivial operations should not be ignored completely. For instance, a string constant containing a complex XML configuration file is highly specific and its reuse a strong indicator for functional similarity.

Thus, the suitability of modules or groups of modules to determine functional reuse needs further investigation. For scalability and practicability, a general and automated approach has to be found to distinguish such cases. One solution could use TF/IDF scores [75] to assign weights to modules based on their usage frequencies. The Zipf-like distribution of module frequencies suggests such an approach.

Workflow Interconnections. Our analysis focussed on characterizing modules that are re-used. An equally valid view is to ask how large the overlap in modules is

for this purpose.

between two workflows. 769 of our 898 workflows share one or more modules with at least one other workflow. On average, each workflow is thus connected to 92.12 other workflows by the use of at least one common module, averaging at 7.23 distinct modules being shared between two workflows. This is remarkable given the findings from Section 3.2.1, as the majority of modules is shown to only be present in single workflows. Apparently, the remaining one third of modules interconnect workflows quite densely.

Figure 3.10 shows how workflows cluster by shared modules. In the figure, two workflows are connected if they share at least three modules. The figure clearly shows clusters of highly interconnected workflows. We manually studied a sample of these clusters and found them to be highly similar in function.

User assistance. The fact that module reuse is uncommon across authors could be interpreted in several ways. One explanation could be that authors are simply not aware (enough) of other people's dataflows and workflows. This situation could be alleviated by using the repository for providing better support for designing scientific workflows [21]. Some work has started already in this direction; for instance, [92] presents a system which recommends web-services for use during workflow design. Accomplishing such functionality for other types of modules and even for subworkflows is highly desirable.

3.4 Summary

This chapter introduced the first study performed on reuse of scientific workflows which has considered reuse at various levels of granularity (module, dataflows, and workflows), at various categories of modules, and also differentiated by workflow authorship. Thereby, we provided three major contributions.

First, we introduced and compared different **methods to identify** modules, dataflows, and whole workflows which we deem suitable for detection of reuse. Our methods allows us to provide fine-grained analyses, and to distinguish functionally important cases of reuse from trivial ones.

Second, our study is the first to consider **authorship**. This allowed us to characterize different kinds of users depending on their usage of the repository, ranging from single time 'authors' uploading duplicate workflows when following a tutorial, to advanced authors creating many functionally interlinked workflows. An important observation obtained by entering this level of detail is that while 36% of workflow elements are reused, only 11% of workflow elements are used by more than one author. Cross-author reuse for dataflows is even lower at 3%. This calls for actions to make authors more aware of the repository contributions by others.

Third, our study investigated **reuse and duplication of workflow elements** in more detail than ever before. Using a categorization of modules helped to better characterize re-use in terms of the types of modules that are reused. Furthermore, it showed that the appearance of single modules in multiple workflows is not per se an

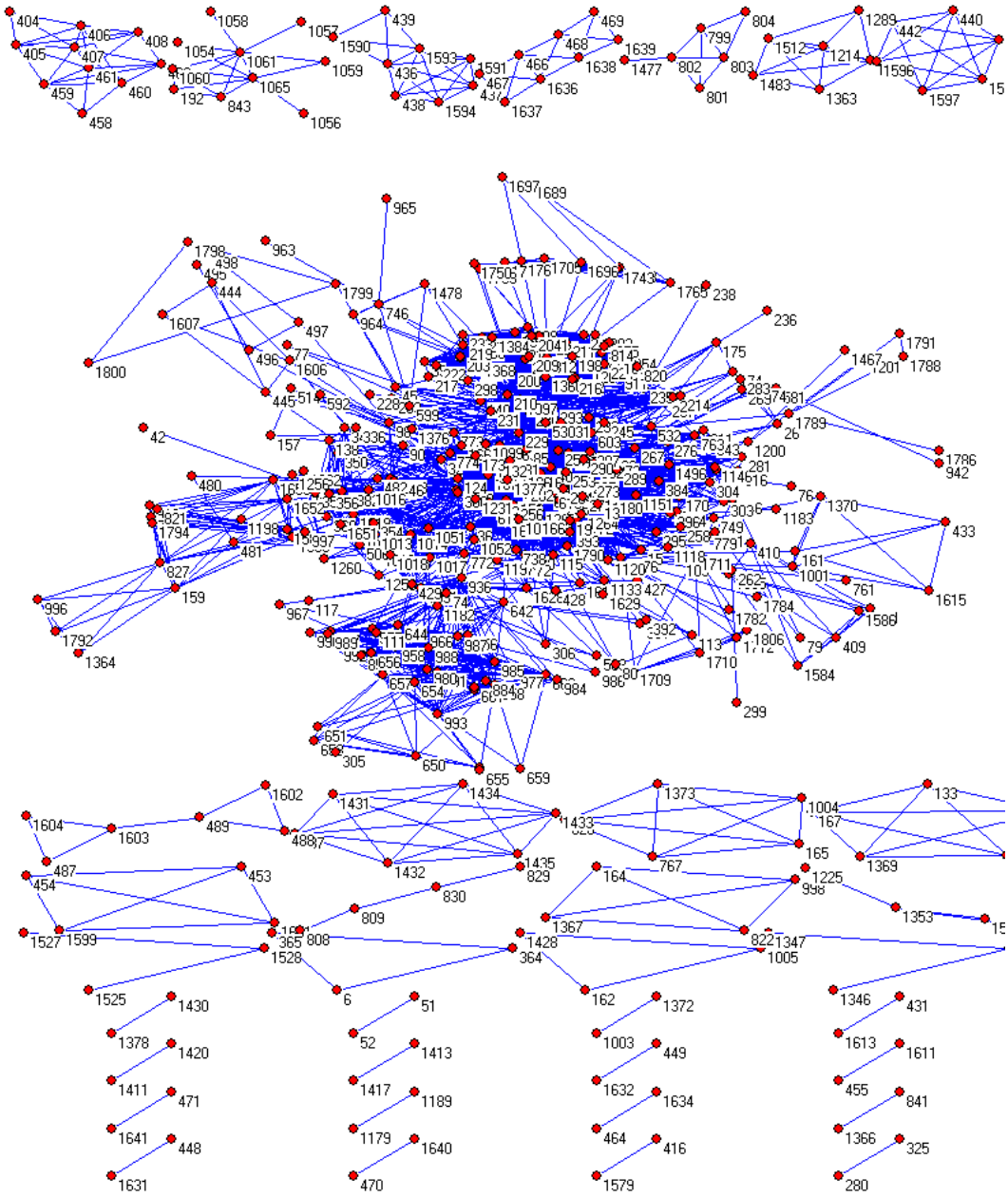


Figure 3.10: Excerpt of a network of workflow-workflow interconnections by at least 3 mutual modules. Labels are myExperiment workflow ids.

indication of functional similarity, and that not all modules are equally well suited for deriving information about functional workflow similarity.

We believe that our findings are important for future work in scientific workflow similarity.

4 A Benchmark for Scientific Workflow Similarity Search

In this chapter, we comparatively investigate and comprehensively evaluate various existing methods for scientific workflow comparison. To do so, we first define a set of interconnected subtasks of which the process of scientific workflow comparison is comprised, and dissect each of the investigated approaches wrt these subtasks. We base our evaluation on an extensive, manually collected gold-standard of similarity ratings over almost 1500 workflows from the myExperiment repository.

Figure 4.1 shows two exemplary scientific workflows from the myExperiment repository. Recall that scientific workflows typically model a dataflow with a structure resembling a directed acyclic graph (DAG) (see Section 2.1.1). They have global inputs and outputs, data processing modules which operate on the data, and datalinks which connect the former and thereby define the flow of data from one module to the next. Each module has attributes associated with it, such as a label, input and output signatures, the type of operation to be carried out, and, if applicable, a set of static, data independent parameters, such as the url of a web-service to be invoked. Upon upload to a repository, workflows typically are further annotated with a title, a general description of their functionality, keyword tags, and the uploading author.

As outlined in Section 2.3.2, existing approaches to similarity measures for these scientific workflows can be classified as being either annotation-based or structure-based, depending on which of the information described above they use. Each of these classes of algorithms has its particular strengths and weaknesses: Annotation-based approaches are independent of the workflows' formats and can be used to compare workflows both across different SWFM, and across multiple repositories [25]. Yet, they only work if the workflows under comparison have annotations, which may or may not be the case for a workflow stored in a public repository by an arbitrary user. Approaches for structural workflow comparison, on the other hand, can be applied without such backing human-provided textual knowledge. Yet, they have to assess a workflow's functionality from the information contained in its DAG structure and the modules it is composed of. As another source of data, provenance represents concrete execution traces of workflows. Such traces would allow, for instance, to take execution parameters and runtime information into account as an additional means of comparison. However, specialized provenance databases have just started to emerge (e.g., in the ProvBench initiative [2]), and we are not aware

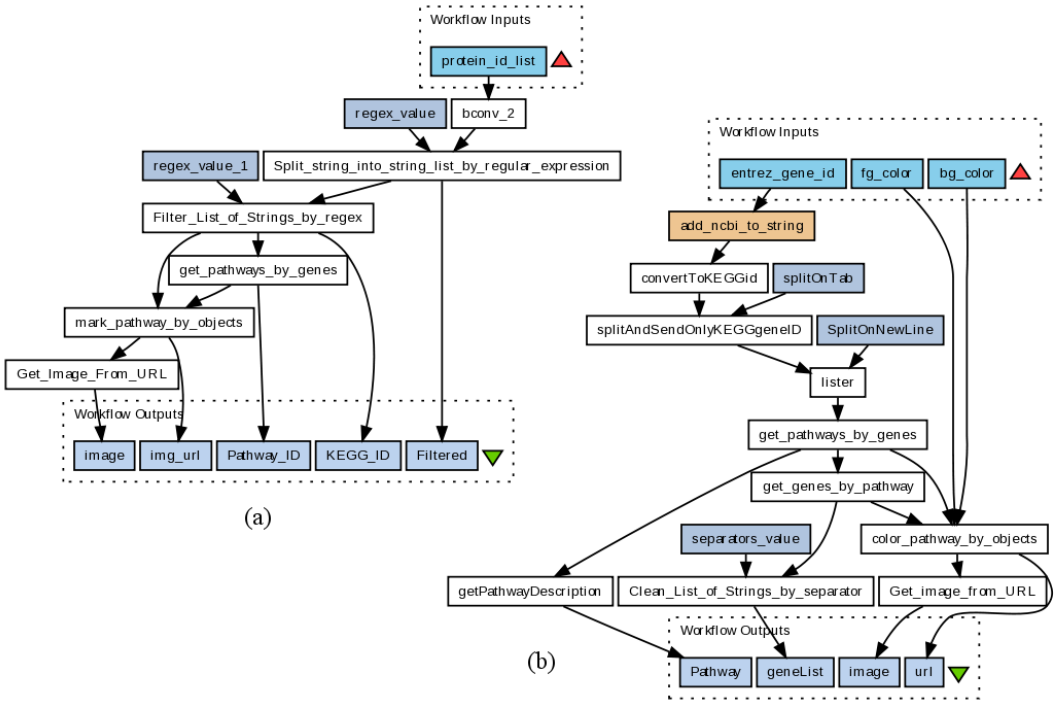


Figure 4.1: Sample scientific workflows from myExperiment: (a) ID: 1189, Title: *KEGG pathway analysis*, (b) ID: 2805, Title: *Get Pathway-Genes by Entrez gene id*.

of any workflow repository also containing real execution traces.

Which approach to scientific workflow comparison provides best results, and how different aspects of workflows contribute to their functional similarity is still an open question. There have been numerous studies investigating both annotational [25, 82, 39] and structural [78, 76, 82, 8, 43, 39, 88] approaches, but their comparison is hindered by a number of factors. Firstly, as shown in Chapter 3, the process of scientific workflow comparison entails several steps from comparison of single modules to comparison of whole workflows [8] - each of which may be treated differently in the methods considered. This makes it hard to determine how single aspects contribute to workflow similarity, and which approach to a specific step of the comparison process provides best results. Secondly, the evaluation of a proposed method is often done by manual inspection of the methods concrete output, or on a proprietary dataset, both hampering repeatability. To compare multiple methods and configurations it is necessary to have a method-independent, gold-standard corpus to evaluate on. To the best of our knowledge, for scientific workflow similarity a gold-standard corpus of decent size does not exist, yet. Reference corpora exist for business process models [33, 34, 32] but these cannot be used easily be-

cause (1) they often come without gold standard ratings and (2) business workflows typically contain rich control structures calling for other similarity measures than purely data-driven scientific workflows [61, 90]. Other work uses synthetic workflows to test similarity measures (e.g.,[51]), while we focus on real-life workflows. Thirdly, the presented evaluations vary with the underlying use case. For instance, similarity measures are a requirement for both clustering and similarity search. The results derived from the corresponding evaluations are difficult to compare.

Addressing these issues, we here present the results of a comprehensive re-implementation and evaluation effort for similarity ranking and retrieval of scientific workflows. Specifically, we make the following contributions:

1. We introduce a conceptual framework that clearly separates the various tasks of workflow comparison, and use it to re-implement a comprehensive set of existing similarity measures.
2. We present an expert-generated corpus of over 2000 similarity ratings for scientific workflows contributed by 15 scientific workflow experts from four international groups - an effort which, to the best of our knowledge, has not been made public before at this extent.
3. We evaluate several algorithms, both annotational and structural ones, on the collected corpus of similarity ratings, showing how each of their steps contributes to the algorithms' quality, and repeat previous experiments where possible. We also investigate how different similarity measures can be successfully combined in ensembles.
4. We additionally investigate how knowledge derived from the repository as a whole can be applied to structural workflow comparison, and show that these modifications benefit result quality or reduce computational complexity, or even both.

In the following, we first introduce our framework for workflow comparison and, reviewing different published methods in detail, show how they were implemented in this framework. In Section 4.2 we give an overview of previous findings on assessing workflow similarity measures. Our experimental setup, including creation of the gold standard corpus, is described in Section 4.3, followed by presentation of our evaluation results in Section 4.4. We conclude in Section 4.5.

This work appeared in [79].

4.1 A Framework for Scientific Workflow Similarity

The functionality of a scientific workflow is determined by the data processing modules it is composed of, and how these modules are connected by datalinks. While

we are ultimately interested in comparing whole workflows, each module represents a distinct functional entity in its own right. As all previous work, we make the reasonable assumption that modules are deterministic and two identical instances of a module are functionally equivalent. Yet, any two different modules may carry the same or very similar functionality. For instance, two different web-services, or a web-service and a locally invoked script, may be functionally equivalent. Thus, to compare scientific workflows wrt their functionality, similarity has to be established on two levels: the level of single modules and the level of whole workflows.

Following this dichotomy, the process of workflow comparison can be conceptually divided into a series of interdependent subtasks: First, the similarity of each pair of modules from two workflows has to be determined by *pairwise module comparison*. Second, using these pairwise module similarities, a *mapping of modules* onto each other needs to be established. Third, the established mapping is used for *topological comparison of the two entire workflows*. Finally, *normalization* of the derived similarity value wrt the sizes of the compared workflows may be desirable.

Note that this setup is not unrelated to that of relational schema matching [72], where mappings between attributes and relations are established. Yet, the elements compared and the underlying intention are quite different: While schema matching compares attributes and relations to establish (mostly) one-to-one equivalences, we here use mappings between modules based on their attributes to derive similarity of entire workflows. Workflow modules are also much richer objects compared to attributes in a schema. Finally, in contrast to schemas workflows have a direction (from source to sink) that is functionally important and that could be exploited for similarity assessment.

Figure 4.2 drafts the comparison process in the context of our similarity framework. This framework implements the identified steps and allows to uniformly apply and combine them. When two workflows are submitted to the framework, their structure and annotations are separated and made available to the corresponding methods of comparison. Before the actual comparison is done, preprocessing of the underlying data may be applied. For annotations, such preprocessing includes the removal of stop words from the workflow descriptions. Using external knowledge derived from the repository, the workflows' structure can be preprocessed in a similar manner. We will explore the application of such repository-derived knowledge more closely in Section 4.1.1. As for the structural comparison process, we refine the task of *topological comparison* by preceding it by a step of *topological decomposition* of the workflows suitable for the intended comparison. This is useful, for instance, when topological comparison is based on substructures of workflows, e.g., subgraphs or paths. The workflows will then first be decomposed into the respective substructures, on which the *module mapping* is performed. The framework is completed by the option of using either a single similarity measure or an ensemble of two or more measures to derive the overall workflow similarity.

In the following, we look at each of the proposed steps of workflow comparison and how they are approached in previous work more closely. An overview on previous

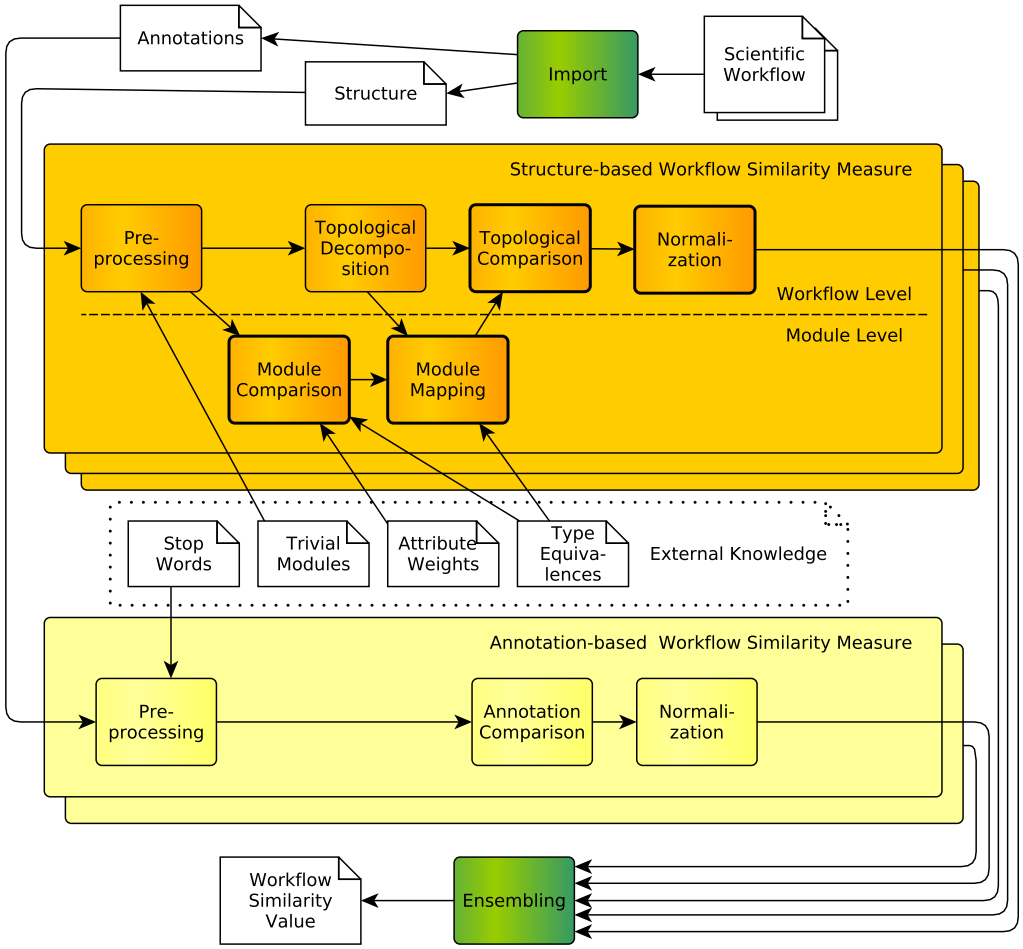


Figure 4.2: Scientific workflow similarity framework

work reflecting the diversity of approaches taken so far can be found in Table 4.1. We restrict our study to methods targeting the modules, structure and/or annotations of workflows as this is the type of data that can be found in current repositories. While approaches making use of workflow provenance or rich semantic annotations on workflows and modules have been studied in related areas, e.g., [6, 12, 15], workflows currently found in public repositories are typically not associated with such data.

4.1.1 Structure-based Measures

Pairwise Module Comparison

In order to apply any of the structural methods for comparing whole workflows, a way of comparing the workflows' elements is needed. Previous work largely relied on module identification by matching of labels [76, 43, 88], i.e., the names given

Table 4.1: Existing approaches to scientific workflow comparison and their treatment of comparison tasks.

Ref.	Annotation-based	Structure-based			Normalization
		Module Comparison	Module Mapping	Topological Comparison	
[25]	bag of words				
[82]	frequent tag sets				
		singular attributes	-	frequent module sets	-
[78]		multiple attributes	greedy	sets of modules	$\|V\|$ of smaller workflow
[8]		semantic annotations	maximum weight	sets of modules and edges	$\|V\| + \|E\|$ of query workflow
		label edit distance	maximum weight	sets of modules and edges	$\|V\| + \|E\|$ of query workflow
[76]		label matching	-	vectors of modules	-
		label matching	-	MCS	$\|V\| + \|E\|$ of larger workflow
[43]		label matching	-	MCS	-
		label matching	-	MCS	workflow sizes
[39]		type matching	-	sets of modules	-
		type matching	-	MCS	-
		type matching	-	graph kernels	-
[88]		label matching	-	GED	-

to a specific instance of a module by a workflows author. In workflow repositories with a heterogeneous author base, however, modules are bound to be labeled non-uniformly. One way of dealing with this heterogeneity is to resort to the matching of other, less varying attributes, such as the modules' types [39]. Another option is to compare labels by their Levenshtein edit distance [8]. To take full advantage of the information contained in a modules specification, however, it seems advantageous to compute module similarity based on a variety of attributes associated with each module, as done in [78]. Which attributes are present in a given module largely depends on the type of operation it provides. For instance, the uri of a web-service to be invoked will only be present in modules designed to invoke a web-service but not in modules performing local operations. Thus, for maximum flexibility, both the set of attributes to compare and the methods to compare them by are configurable in our framework, together with the weight each attribute has in computation of overall module similarity. This approach subsumes all previously proposed methods for module comparison by using appropriate configurations. In our evaluation (see Section 4.4), we shall investigate the following configurations *pX* for module comparison:

pw0, used as a default, assigns uniform weights to all attributes, and compares module *type*, and the web-service related properties *authority name*, *service name*, and *service uri* using exact string matching. Module *labels*, *descriptions*, and *scripts* in scripted modules are compared using Levenshtein edit distance [56].

pw3 compares single attributes in the same way as *pw0* but uses higher, but not uniform weights for *labels*, *script* and *service uri*, followed by *service name* and *service authority* in the order listed, similar to [78].

pll disregards all attributes but the *labels*, and compares them using the Levenshtein edit distance, resembling the approach taken in [8].

plm disregards all attributes but the *labels*, and compares them using strict string matching as done in [76, 43, 88].

Module Mapping

After generating all pairwise module similarities, a *mapping* of the modules has to be established. Such a mapping selects the best similarity match between the modules from the two compared workflows. When using strict matching of singular module attributes to derive module similarity [76, 43, 88, 39], such as the labels, the module mapping is implicitly given as the set of matching modules. When module comparison is based on more complex similarity assessment, the best similarity match between the modules of the two compared workflows has to be found explicitly. Previous approaches to this task include greedy selection of mapped modules [78] and computation of the mapping of maximum overall weight [8], both of which have been included in our framework. For clarity of presentation, in the following we only refer to the latter approach of *maximum weight matching (mw)*. We compare both approaches in Section 4.4.1.

Additionally, when an order of the modules to be mapped is given by the *topolog-*

ical decomposition of the workflows, their *maximum weight non-crossing matching (mwnc)* [63] can be determined to take the given order of modules into account. That is, given two orderings of modules $(m_1, ..m_i, ..m_k)$ and $(m'_1, ..m'_j, ..m'_l)$, a mapping of maximum weight is computed where the result cannot contain two mappings (m_i, m'_j) and (m_{i+x}, m'_{j-y}) with $x, y \geq 1$.

Topological Workflow Comparison

Regarding topological comparison of scientific workflows, existing approaches can be classified as either a) structure agnostic, i.e., based only on the sets of modules present in two workflows, [78, 76, 82, 8]; b) based on substructures of workflows, such as maximum common subgraphs [76, 43, 39] or graph kernels derived from frequent subgraphs [39]; or c) using the full structure of the compared workflows [88]. We include an approach to topological comparison for each of these classes. We denote the DAG of a workflow as $G_{wf} = (V_{wf}, E_{wf})$.

Sets of Modules. Analogous to the similarity measure described in [78, 76, 82, 8], two workflows $wf1$ and $wf2$ are treated as sets of modules. The additive similarity score of the module pairs mapped by *maximum weight matching (mw)* is used as the non-normalized workflow similarity $nnsim_{MS}$, with $sim(m, m')$ denoting a module pairs similarity value:

$$nnsim_{MS} = \sum sim(m, m') \mid (m, m') \in mw(V_{wf1}, V_{wf2})$$

Sets of Paths. As a slightly relaxed version of using the maximum isomorphic subgraph for workflow comparison [76, 43, 39], the sets of all paths two DAGs are comprised of can be used to compare them by their maximum similar subgraph [54]. We follow this notion and *topologically decompose* each workflow into its set of paths: Starting from each node without inbound datalinks (the DAGs source nodes), all possible paths ending in a node without further outbound links (the DAGs sink nodes) are computed. All pairs (P, P') from the so collected sets of paths PS_{wf1} and PS_{wf2} are compared using the *maximum weight non crossing matching* scheme $(mwnc)$ to determine the additive similarity score for each pair of paths:

$$sim(P, P') = \sum sim(m, m') \mid (m, m') \in mwnc(V_{wf1}, V_{wf2})$$

To determine the maximum non-normalized similarity of the two workflows wrt their so compared sets of paths, a *maximum weight matching (mw)* of the paths is computed on the basis of these pairwise path similarity scores:

$$nnsim_{PS} = \sum sim(P, P') \mid (P, P') \in mw(PS_{wf1}, PS_{wf2})$$

Graph Edit Distance. Analogous to the work presented in [88], the full DAG structures of two workflows are compared by computing the graph edit distance using the SUBDUE [66] package. SUBDUE allows labels to identify nodes in a graph, which it uses during the graph matching process. To transform similarity of modules to identifiers, we set the labels of nodes in the two graphs to be compared to reflect the module mapping derived from *maximum weight matching* of the modules during conversion of the workflows to SUBDUEs input format.

The workflows' non-normalized similarity is then computed as

$$nnsim_{GED} = -cost_{GED}$$

For computing graph edit distance, we keep SUBDUEs default configuration which defines equal costs of 1 for any of the possible edit operations (as in [32]). Testing several different weighting schemes did not produce significantly different results.

Normalization

Whether or not to normalize the similarity values derived from topological workflow comparison and how it is to be done strongly depends on the intended use case. When, as in our study, the interest is to determine overall workflow similarity, the goal of the normalization step will be to maximize the information about how well two workflows match globally. As an example, consider two sets of two workflows each, containing 2 and 3 modules, and 98 and 99 modules, respectively, compared by graph edit distance. If in both cases the workflows match perfectly, with the exception of 1 module and 1 edge, the graph edit distance will be 2 in both cases. Yet, intuitively, the similarity of the workflows in the second set would be deemed higher. Indeed, experimental evaluation showed that normalization wrt workflow size provides significantly better results (see Section 4.4.1).

The step of normalization has been approached rather heterogeneously in previous work (see Table 4.1). Next to the consideration of workflow size taken therein, our general intention is to acquire similarity values in the range of [0,1]. For set based topological comparisons we resort to a variation of the Jaccard index for set similarity: The Jaccard index is used to express the similarity of two sets A and B by their relative overlap $\frac{|A \cap B|}{|A \cup B|}$ which is equivalent to $\frac{|A \cap B|}{|A|+|B|-|A \cap B|}$. Our modification reflects the fact that mapped elements of our sets of modules or paths are mapped by similarity, not identity. Thus, the size $|V_{wf1} \cap V_{wf2}|$ of the overlap between the two module sets, for instance, is replaced by the overlaps maximum similarity score captured by $nnsim_{MS}$ to derive the overall module set similarity of two workflows:

$$sim_{MS} = \frac{nnsim_{MS}}{|V_{wf1}| + |V_{wf2}| - nnsim_{MS}}$$

The rational here is that where the classical Jaccard index compares the number of mutual elements in the sets with their overall number of (distinct) elements, our modification compares the amount of similarity of similar elements from the sets

with the non-similar remainder. If two workflows are identical, each module has a mapping with a similarity value of 1. Then $nnsim_{MS} = |mw(V_{wf1}, V_{wf2})| = |V_{wf1}| = |V_{wf2}|$, and $sim_{MS} = 1$.

For path sets, the normalization is analogous with $|PS_{wf}|$ and $nnsim_{PS}$.

For graph edit distance, we normalize the obtained edit cost by the maximum cost possible. This maximum cost depends on the costs assigned to each edit operation. For our configuration of uniform costs of 1, we thus use the following normalization:

$$sim_{GED} = 1 - \frac{cost_{GED}}{max(|V_{wf1}|, |V_{wf2}|) + |E_{wf1}| + |E_{wf2}|}.$$

The rational here is that in the worst case, each node in the bigger set of nodes is either substituted or deleted, while for the edges a complete set of insertions and deletions may be necessary.

Including Repository Knowledge

Knowledge derived from a repository of scientific workflows [87, 81], can be used in the structural comparison process. We investigate two possible applications of such knowledge.

Module Pair Preselection. When comparing sets of modules V_{wf1} and V_{wf2} from two scientific workflows, the general approach is to compare each pair of modules from the Cartesian product $V_{wf1} \times V_{wf2}$ of the workflows' module sets. To reduce the amount of module pair comparisons, restrictions can be imposed on candidate pairs by requiring certain module attributes to match in a certain sense. As modules of the same *type* are more likely to carry similar functionality, a first strategy for candidate selection requires module *types* to match. As a second, less strict selection strategy, we cast module *types* to equivalence classes based on the categorization proposed in [87]. For instance, one such class holds all web-service related module types. Modules within each such class may be compared and mapped onto each other. This introduction of *type equivalences* was motivated by the observation that in the used dataset of Taverna workflows, especially web-service modules are typed with a variety of identifiers, such as 'arbitrarywsdl', 'wsdl', or 'soaplabwsdl'.

Importance Projection Preprocessing. Not all modules in a scientific workflow contribute equally to the workflows specific functionality: Modules used most frequently across different workflows often provide trivial, rather unspecific functionality, such as splitting a string into a list of strings (see Section 3.2.1). To account for this, we assign a score to each module indicating the importance of the module for a workflows specific functionality. Only modules with a score above a configurable

(a) (b)

Figure 4.3: Sample importance projection of scientific workflow 1189 (a) and 2805 (b) (see also Fig. 4.1).

threshold are kept for further use in module and workflow similarity computation. The workflow is thus projected onto its most relevant modules. In order to make full use of this projection in all our structural similarity measures, all paths between important modules are preserved as edges in terms of the transitive reduction of the resulting DAG. That is, if two important modules are connected by one or more paths along non-important modules in the original workflow, they are connected by one edge in the *Importance Projection (ip)*. Figure 4.3 shows the resulting projection of our example workflows 1189 and 2805. The selection of important modules is currently done manually based on module types. Modules that perform predefined, trivial local operations are removed.

4.1.2 Annotation-based Measures

Purely annotation-based methods use only textual information recorded with the workflows in a repository. Such information includes the workflow's title, a free form text description, and assigned keyword tags. Two approaches to annotational similarity of scientific workflows have been proposed which we include in our framework:

Bag of Words. Following the work of [25], workflows are compared by their titles and descriptions using a bag-of-words approach. Both title and description are tokenized using whitespace and underscores as separators. The resulting tokens are converted to lowercase and cleansed from any non alphanumeric characters. Tokens are filtered for stopwords. The workflows' similarity is then computed as

$$sim_{BW} = \frac{\#matches}{\#matches + \#mismatches}$$

where $\#matches$ is the number of tokens found in both workflows' title or description, and $\#mismatches$ is the number of tokens present only in either one workflow.

This quotient again corresponds to the Jaccard index for set similarity. Please note that our algorithm presented here does not account for multiple occurrences of single tokens. We did try variants that do so, but evaluation showed that these variants performed slightly worse than the one described here (data not shown).

Bag of Tags. The keyword tags assigned to scientific workflows in a repository can also be used for similarity assessment, as done in [82]. The tags assigned to a workflow are treated as a bag of tags and calculate workflow similarity in the same way as in the *Bag of Words* approach described above. Following the approach of [82], no stopword removal or other preprocessing of the tags is performed. This also aligns with our expectation of tags to be specifically preselected by the workflow author.

4.2 Previous Findings

After the in depth review of existing approaches to the various steps of scientific workflow comparison in the previous section, we here summarize results of all previous evaluations we are aware of. The concrete type of evaluation the proposed methods were subjected to varies with the intended use case, e.g., clustering or similarity search. Additionally, evaluation settings vary greatly from manual inspection of a methods output [25, 82, 78, 43] to systematic evaluation against a small (and not publicly available) set of expert provided truth [8] or against ground truth derived from another method [39]. The following findings have been reported:

Module Comparison. [8] found that similarity based on semantic annotations of workflow modules provides better results in workflow retrieval than using module similarity based on edit distance of their labels. Note that in this work, semantic annotations were manually assigned to modules, whereas scientific workflows in public repositories usually don't contain such semantic information. Comparing modules by matching of labels, [43] found the lowercasing of labels to slightly improve ranked retrieval on the example of a single query workflow.

Topological Comparison. [78] found workflows regarded similar by comparison of their sets of modules to also be structurally similar upon manual inspection, indicating a correlation between the modules used in a workflow and its topology. Similarly, comparing the use of maximum common isomorphic subgraphs (MCS) and module label vectors for workflow comparison, [76] found that both methods deliver similar results. Conversely, [39] found MCS to perform better than bags of modules, and both of them to be slightly outperformed by graph kernels.

Normalization. [43], as the only study we are aware of to provide comparative information about this aspect, suggests that normalization wrt workflow size improves

results of scientific workflow comparison.

Annotational Comparison. Approaches based on workflow annotations have been generally reported to deliver satisfying results [25, 82, 39]. They have been found to perform either as good as or better than structural approaches in workflow comparison [82, 39].

4.3 Experimental Setup

The goal of this work is an unbiased and comprehensive benchmark of different methods for similarity search in a corpus of scientific workflows. To this end, we re-implemented all methods described in Section 3, structured according to the steps of our comparison framework. We collected a quite comprehensive corpus of similarity ratings for a set of Taverna workflows from the myExperiment scientific workflow repository [74]. A second set of workflows was assembled from the Galaxy online repository [45]. Gold standard ratings were obtained from 15 field experts and aggregated using median ranking. We compare all algorithms regarding *ranking correctness* and *retrieval precision*. These topics are discussed in more detail in the following. Note that all data are made publicly available.[1]

4.3.1 Workflow Dataset

myExperiment provides a download of all workflows publicly available in its repository. For Taverna workflows, which make up approx. 85% of myExperiment [81], this dataset contains an RDF representation of the workflow structure, along with attributes of the modules used and annotations provided by authors. We transformed all workflows into a custom graph format for easier handling. During this transformation, subworkflows are inlined and input and output ports were removed. The complete dataset contains 1483 workflows. myExperiment - and thus our dataset - is generally domain agnostic. To match the expertise of our expert curators, we focussed on workflows from the life sciences domain in the evaluation. We will discuss possible implications of this decision in Section 4.5.

To investigate transferability of findings to other workflow formats, we also created a secondary eval data set of 139 Galaxy workflows from the public Galaxy repository. These workflows were transformed into the same internal format as described above and processed using the exact same methods. We performed most experiments with the larger myExperiment data and used the Galaxy data set as independent validation and to study data set specific properties and their implications on algorithm performance (see Section 4.4.3).

[1] https://www.informatik.hu-berlin.de/forschung/gebiete/wbi/resources/flowalike.

4.3.2 Expert Ratings

We conducted a user study to collect manually assigned similarity ratings for selected pairs of scientific workflows. Overall, 15 domain experts from six different institutions participated. Ratings were obtained in two phases:

In a first experiment, the goal was to generate a corpus of ratings independent of a concrete similarity measure to make it suitable for evaluation of large numbers of different measures, and measures to be developed in the future. 24 life science workflows, randomly selected from our dataset, (called *query workflows* in the following) were presented to the users, each accompanied by a list of 10 other workflows to compare it to. To obtain these 10 workflows, we ranked all workflows in the repository wrt a given query workflow using a naive annotation based similarity measure and draw workflows at random from the top-10, the middle, and the lower 30. The ratings were to be given along a four step Likert scale [58] with the options *very similar*, *similar*, *related*, and *dissimilar* plus an additional option *unsure*. *Unsure* user ratings were not further considered in the evaluation. The ratings collected in this first experiment were used to rank the 10 workflows for each of the query workflows. The individual experts' rankings were aggregated into consensus rankings using the BioConsert algorithm [20], extended to allow incomplete rankings with *unsure* ratings. On the basis of the generated consensus rankings, we evaluate the algorithms' *ranking correctness* in Section 4.4.1. Figure 4.4 inspects inter-annotator agreement, comparing each single expert's rankings to the generated consensus using the ranking correctness and completeness measures described below in section 4.3.3. While we do see a few outliers, most experts are rather d'accord about how workflows are ranked.

In a second experiment, the selected algorithms were run to each retrieve the top-10 similar workflows from our complete dataset of 1483 Taverna workflows for eight of the 24 query workflows from the first experiment. The results returned by each tested algorithm were merged into single lists between 21 and 68 elements long (depending on the overlap in the algorithm's top-10). Naturally, these lists did contain workflows already rated in the first experiment. Experts were now asked to complete the ratings using the same scale as before. The ratings provided in this second experiment qualify each workflow in the search results of each of the used algorithms wrt their user-perceived similarity. Using these completed ratings we evaluate the algorithms' *retrieval precision* in Section 4.4.2 (definition below). Different experts' opinions were aggregated as the median rating for each pair of query and result workflow.

Altogether, we presented 485 workflow pairs (24 x 10 from the first experiment and 255 non-overlapping pairs from the second) to the 15 experts and received a total of 2424 ratings.

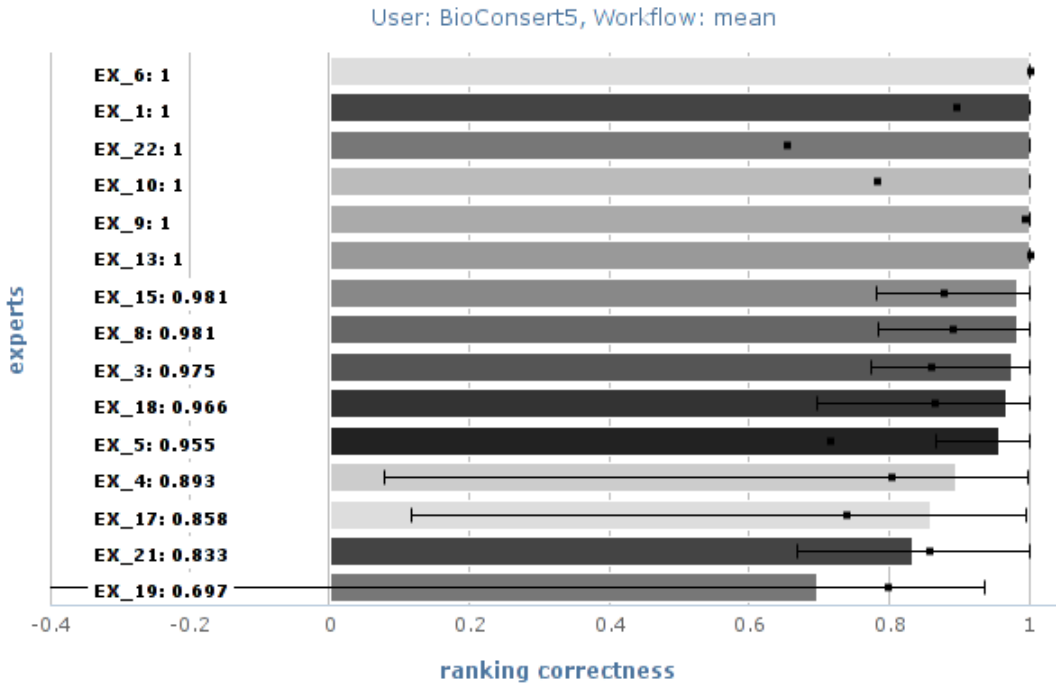


Figure 4.4: Mean ranking correctness (bars) with upper and lower stddev (error-bars), and mean ranking completeness (black squares) for single experts' rankings compared to the ranking derived as BioConsert expert consensus.

4.3.3 Evaluation Metrics

As proposed in previous work [8], we use the measures of ranking *correctness* and *completeness* [19] to compare the algorithms' rankings against the experts' consensus rankings in our first experiment. To evaluate the algorithms' retrieval performance on the ratings from our second experiment, we compute *precision at k* of the algorithmic search results.

Ranking Correctness. For ranking correctness, the order of each pair of elements in the rankings is compared. If in both rankings the elements are not tied and their order is the same, the pairs order is called *concordant*. If their orders differ, the pair is *discordant*. Pairs tied in either of the two rankings do not count for correctness, which is computed as

$$correctness = \frac{\#concordant - \#discordant}{\#concordant + \#discordant}$$

where $\#concordant$ and $\#discordant$ are the numbers of concordant and discordant pairs, respectively. Correctness values range from -1 to 1, where 1 indicates full

correlation of the rankings, 0 indicates that there is no correlation, and negative values are given to negatively correlated rankings.

Ranking Completeness. Ranking completeness, on the other hand, measures the number of pairs of ranked elements that are not tied in the expert ranking, but tied in the evaluated algorithmic ranking.

$$completeness = \frac{\#concordant + \#discordant}{\#pairs\_ranked\_by\_experts}$$

The objective here is to penalize the tying of elements by the algorithm when the user distinguishes their ranking order.

Retrieval Precision at k. The algorithms' retrieval precision at k is calculated as

$$P@k(result\ list) = \frac{1}{k}\sum_{i=1}^{k} rel(r_i)$$

at each rank position $0 < k \leq 10$, with $rel(r_i) \in 0, 1$ being the relevance of the result at rank i. As the expert ratings on whether a workflow is a relevant result for similarity search with a given query workflow are quaternary by the Likert scale used, we consider different relevance thresholds: *very similar*, *similar* or *related*. For instance, only workflows with a median rating of *similar* are regarded to be relevant.

4.4 Results

We now present the results obtained from the two experiments on algorithmic *workflow ranking* and *workflow retrieval*. We start with a baseline evaluation, focussing on the methods of comparing workflows proposed in previous work: *Sets of Modules* [78, 76, 82], workflow substructures [76, 43, 39] in terms of their *Sets of Paths* [54], *Graph Edit Distance* [88], *Bags of Words* [25], and tag-based workflow comparison [82, 39] as *Bags of Tags*. We then investigate each step of workflow comparison in detail, from different approaches to module comparison to the inclusion of external knowledge in the comparison process. As we will see, each of these steps plays a defining role for result quality, and using the right settings can significantly improve result quality.

4.4.1 Workflow Ranking

Baseline Evaluation

Figure 4.5 shows mean ranking correctness and completeness over all query workflows for each of the similarity algorithms under investigation. Note that for *Graph Edit Distance*, we allowed match cost computation of each of the 240 pairs of scientific workflows to take a maximum of 5 minutes. 23 pairs not computable in this

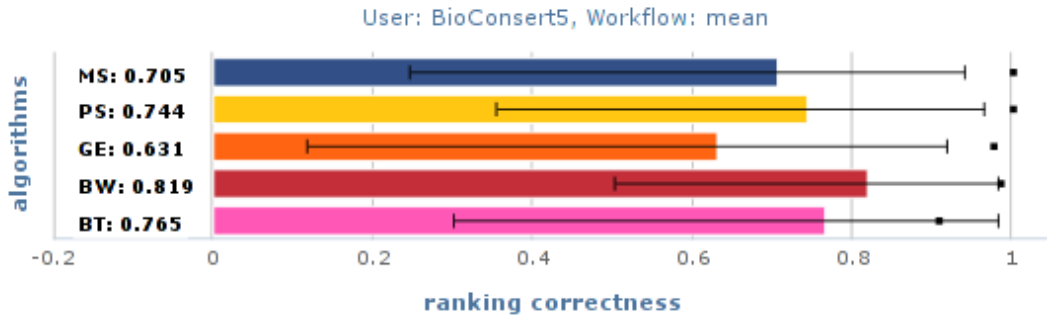


Figure 4.5: Mean ranking correctness (bars) with upper and lower stddev (error-bars), and mean ranking completeness (black squares) for similarity algorithms against BioConsert expert consensus. Numerical values denote mean average correctness.

timeframe were disregarded in the evaluation. All algorithms are used in their basic, normalized configurations with uniform weights on all module attributes. The sim_{BW} algorithm has the best results in terms of ranking correctness. sim_{BT} and the structural similarity measure sim_{PS} almost tie, followed by sim_{MS}. sim_{GE} delivers worst performance and is the only algorithm in this set with a statistically significant ($p<0.05$, paired ttest) difference to sim_{BW}. These findings largely confirm those of previous work (see Section 4.2) that annotational measures outperform structural workflow similarity — in certain settings. The good performance of sim_{BW} is not surprising: Well written titles and descriptions capture the main functional aspects of a workflow, and provide a strong means for judging workflow similarity.

Interestingly, while most structural similarity measures are complete in their ranking, both annotational measures tie some of the compared workflows where experts see them differently ranked. This is especially true for sim_{BT}, which, additionally, is not able to provide rankings for four of the given query workflows due to lack of tags. These workflows were not considered for computation of sim_{BT} ranking performance. Note that around 15% of the workflows in our complete dataset lack tags.

As for the different structural comparison methods, sim_{GE} is clearly outperformed by the other two. This indicates that many functionally similar workflows, while sharing common modules and substructures, do differ in their overall graph layout. We will see this observation confirmed in Section 4.4.1, when workflows are preprocessed to remove structural noise.

With these initial evaluation results as a baseline, we inspect several aspects of workflow comparison for their impact on ranking performance.

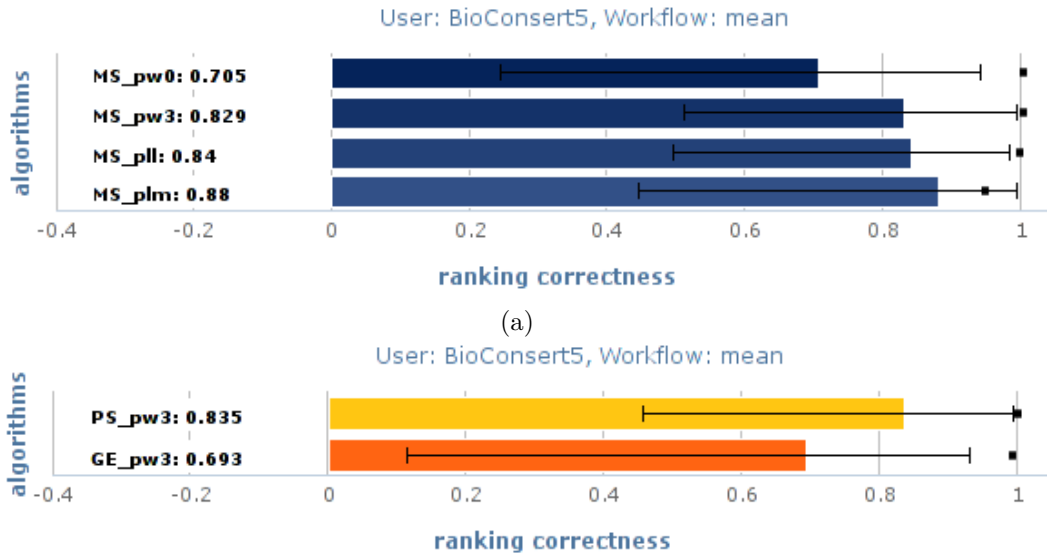
Table 4.2: Algorithm shorthand notation overview

Notation	Description
MS	*Module Sets* topological comparison
PS	*Path Sets* topological comparison
GE	*Graph Edit Distance* topological comparison
BW	*Bag of Words* annotation based comparison
BT	*Bag of Tags* annotation based comparison
np	No structural preprocessing of workflows
ip	*Importance projection* workflow preprocessing
ta	No module pair preselection for comparison
te	*Type equivalence* based module pair preselection
pw0	Module comparison with uniform attribute weights
pw3	Module comparison on tuned attribute weights
pll	Module comparison by edit distance of labels only
plm	Module comparison by matching of labels only

Module Comparison (pX)

Figure 4.6a shows the impact of the different module similarity schemes (see Section 4.1.1 and Table 4.2 for an overview of the used notation) on ranking correctness by the example of the sim_{MS} algorithm. Trends are similar for the other measures (data not shown). Clearly, the uniform weighting scheme *pw0* used in the baseline evaluation of Figure 4.5 performs worst ($p<0.05$). Using only the edit distance of *labels* for module comparison (*pll*) is on par with the more complex scheme *pw3* (using refined weights on various attributes) in terms of mean ranking correctness. Note that there is a minimal reduction in ranking completeness for *pll*, resulting from the less fine grained similarity assessment. This reduction in completeness is much more pronounced when using label matching (*plm*) for assessing module similarity. Further investigation showed that the striking increase in ranking correctness *plm* achieves is in fact due to this reduction in ranking completeness, as pairs of workflows tied by the algorithmic ranking are not accounted for when determining ranking correctness: while the most similar workflows are ranked high, less similar ones are not distinguished in terms of their similarity any more.
The (to us) surprisingly good performance of the *label* only approaches, especially *pll*, shows that while the author base of the workflows in our dataset is heterogeneous, the labels given to modules mostly do reflect their function and thus convey measurable information about their similarity. Yet, the strict matching of labels, as used in many previous studies, offers too little fine grained similarity for differentiated assessment of workflow similarity. We will see how this observation affects workflow retrieval in Section 4.4.2.

The change in the algorithms' ranking performance induced by both the *pw3* and *pll* weighting schemes (see Fig. 4.6a and b) puts these algorithms ahead of sim_{BW},



(a)

(b)

Figure 4.6: Mean ranking correctness for (a) sim_{MS} with various module attribute weightings, and (b) sim_{PS} and sim_{GE} with $pw3$.

if yet not significantly. The exception here is sim_{GE}, where the impact of different weighting schemes is less pronounced. The reason for this most probably lies in the specific type of structural comparison applied: The more overall workflow topology is taken into account, the less do differences in similarity assessment of single pairs of modules matter.

From here on, we denote the module weighting scheme used with an algorithm by specifying the corresponding pX in its name (e.g., *GE_pw3* refers to the sim_{GE} using the $pw3$ module attribute weighting scheme for module comparison).

Module Mapping and Normalization

We investigated the impact of different module mapping strategies and of using normalization or not. Figure 4.7 exemplifies our findings on two settings that have been used in previous approaches: (1) Greedy mapping of modules in the *Module Sets* measure [78] has no impact on ranking quality when compared to using maximum weight matching (see Fig. 4.5). Remarkably, this indicates that in the set of workflows studied, the mappings of modules are rather nonambiguous, i.e. modules mostly have a single best mapping partner in the respective other workflow. (2) The omission of normalization of similarity values with *Graph Edit Distance* [88], on the other hand, significantly reduces ranking correctness ($p<0.05$) when compared to results for normalized sim_{GE} (see Fig. 4.5). This data confirms the results from [43], but on a much larger data base.

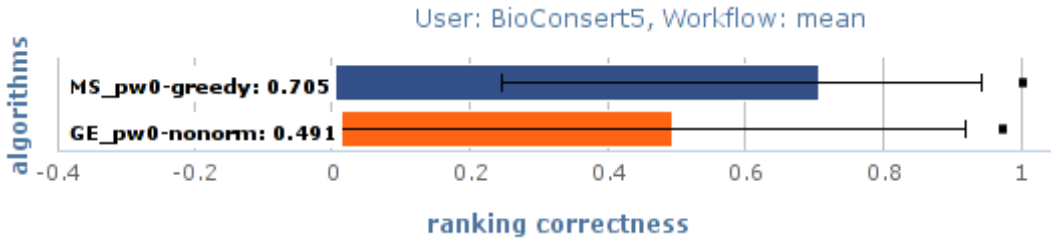


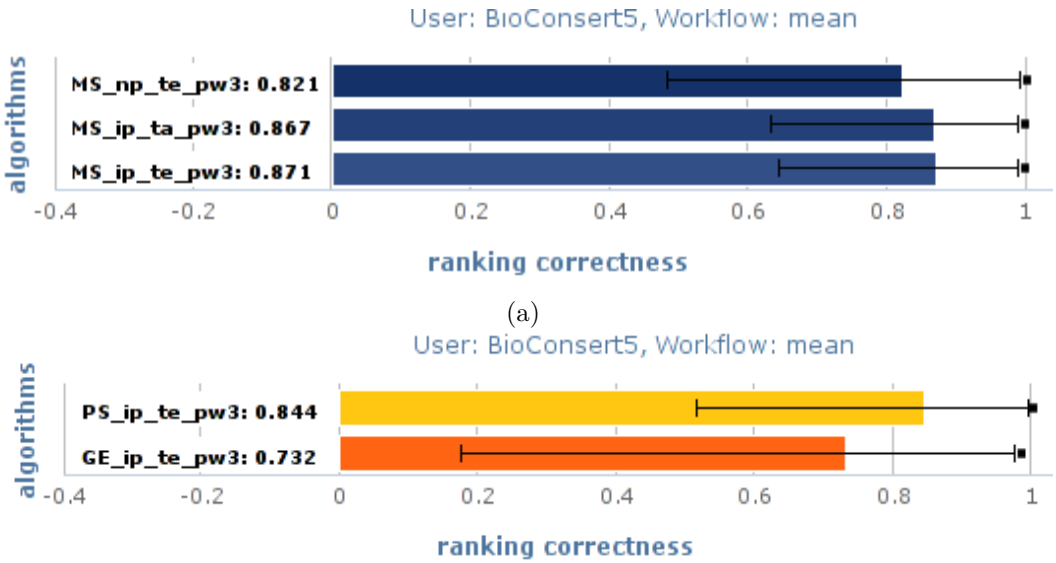
Figure 4.7: Mean ranking correctness for sim_{MS} with greedy mapping of modules and sim_{GE} without normalization of edit distance.

Including Repository Knowledge

Module Pair Preselection (tX). For sim_{MS}, Figure 4.8a (first bar) shows the impact of the *type equivalence (te)* module pair preselection strategy proposed in Section 4.1.1. The trends of these changes are similar for all algorithms: While strict type matching significantly decreases the algorithms' ranking correctness (data not shown), the use of equivalence classes results in correctness values comparable to those without any restrictions on module pair selection. This is remarkable, as it shows that a) the technical classes of modules (*web-service, script, local operation*) do play an important role in determining a modules function; and b) this technical distinction of modules is also detected when comparing all pairs of modules - even when using only labels.

While *te* doesn't lead to an improvement of user perceivable ranking quality, the exclusion of certain module pairs from comparison yields a reduction of such pairwise module comparisons by a factor of 2.3 (172k/74k on the evaluation dataset of experiment 1) and thus a notable runtime reduction. From here on, we denote the module pair selection used with an algorithm by specifying '*te*' for equivalence class based selection and '*ta*' for selection of all pairs of modules in the algorithms name.

Importance Projection (Xp). As shown in Figure 4.8a and b, most algorithms benefit from adding *Importance Projection (ip)* preprocessing (Section 4.1.1), with the exception of sim_{PS} showing stable results. The reduction in structural noise is especially visible in sim_{GE}: As different ways of transforming intermediate results within the workflow are removed, similarities in the constellations of the most specific functional modules become more pronounced. This positive impact of *ip* on the correlation of algorithm rankings with expert rankings confirms our intention of removing presumably unimportant modules, and confirms our selection of modules to keep. Yet, such manual selection is only possible with both in depth understanding of the types of modules used in a dataset of scientific workflows, and knowledge about their specific relevance to a workflow's functionality. An interesting line of research to be explored in future work are methods to perform such a preselection automatically, for instance based on module usage frequencies.



(a)

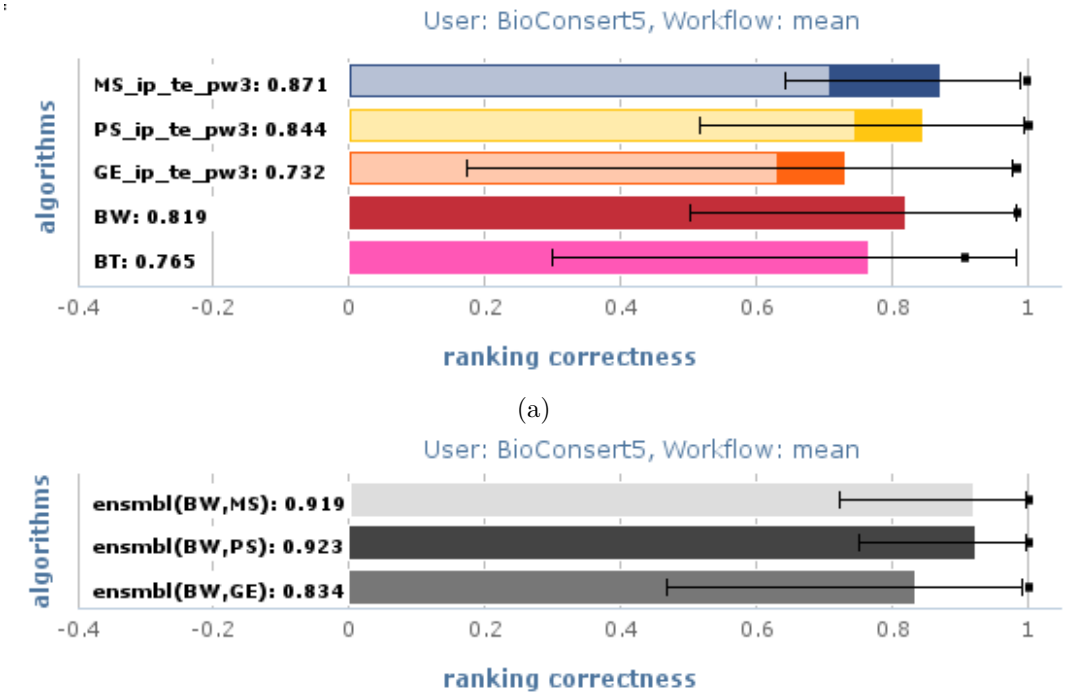
(b)

Figure 4.8: Mean ranking correctness for (a) sim_{MS}, and (b) sim_{PS} and sim_{GE} when including external knowledge.

As a side effect, *ip* leads to a decrease in the average number of modules per workflow from 11.3 to 4.7 in our dataset, resulting in a significant increase in computational performance of all structural algorithms. This effect is especially relevant for *Graph Edit Distance* computation: where sim_{GE} was able to compute the similarities of only 217 of the 240 workflow pairs of our ranking experiment within the per-pair timeout of 5 minutes without using *ip*, it can now compute all pairs but one. From here on, we denote the use of *importance projection* with an algorithm by specifying '*ip*' in its name, and '*np*' for its omission.

Best Configurations

The three modifications *pX (module comparison scheme)*, *tX (module pair preselection)*, and *Xp (importance projection preprocessing)* can be used in all combinations with each of the algorithms, resulting in a total of 72 different configurations (not considering different methods for module mapping and normalization). For each algorithm, Figure 4.9a shows the configuration with its best standalone ranking performance in direct comparison to the annotation-based approaches: When tuned appropriately, algorithms based on workflow structure outperform annotation based approaches, except when very strict graph comparison is applied as in sim_{GE}. Note that the differences between *pw3* and *pll* are minimal and not significant.



(a)

(b)

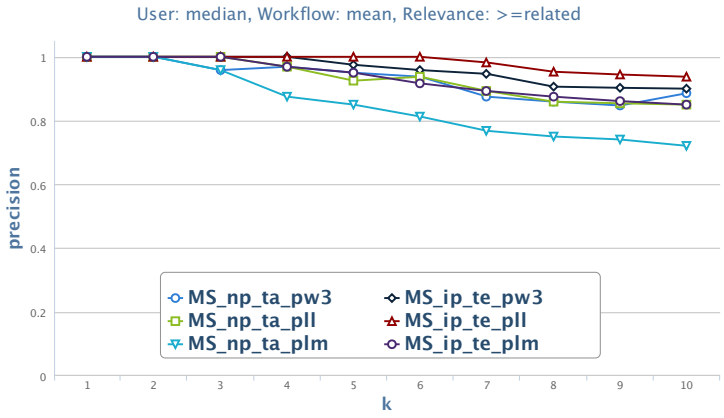
Figure 4.9: Mean ranking correctness for a) single algorithms' best configurations (shaded: baseline evaluation, see Fig. 4.5), and b) the best ensembles of two (see text).

Ensembles of Algorithms

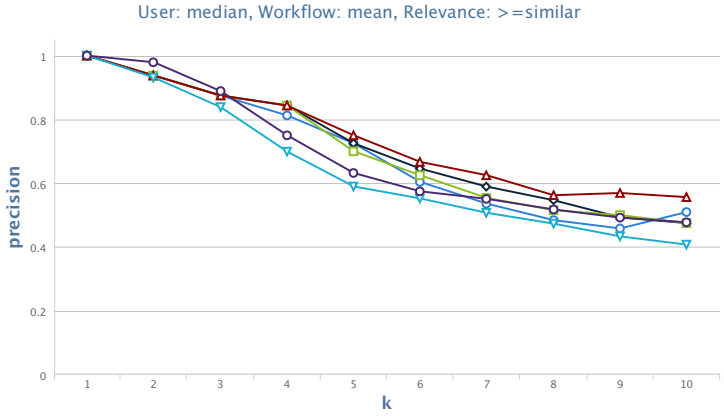
Just as consensus can be generated by aggregating expert rankings, the rankings produced by the similarity algorithms can be combined into a single ranking. We tested such ensembles by simply taking the average of the scores of selected individual ranking algorithms. This especially allows to integrate the different perspectives of annotational and structural workflow similarity. We ran experiments for all combinations of two algorithms and indeed found the best performing ensembles to aggregate sim_{BW} and either sim_{MS} or sim_{PS} with *ip*, *te* and *pll*. The resulting improvement of ranking performance of these ensembles over any algorithm used on its own is both significant ($p < 0.05$) and substantial (see Fig. 4.9b). As implied by the reduction of standard deviation, results are also much more stable.

4.4.2 Workflow Retrieval

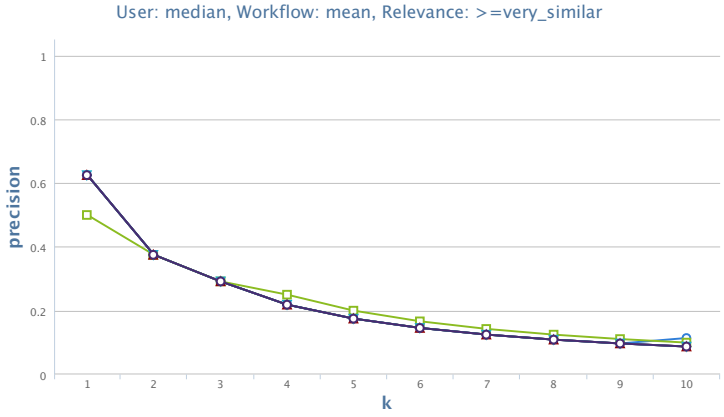
We investigate the algorithms' retrieval performance over a whole repository in terms of *retrieval precision* over the top 10 search results. We found the results of the first experiment mostly confirmed and only report the most interesting findings here. We first inspect results for different module similarity schemes, followed by a comparison



(a)

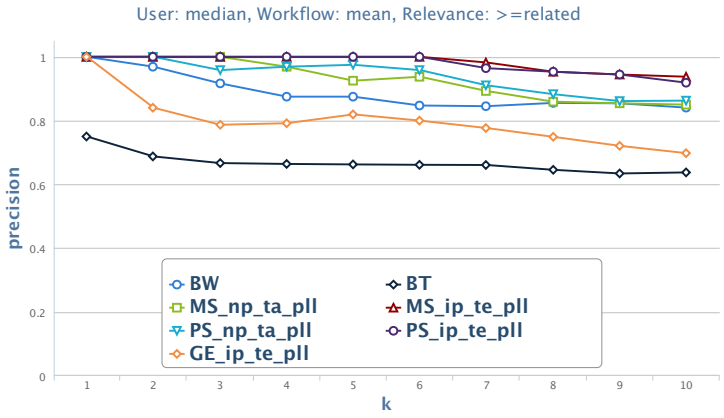


(b)

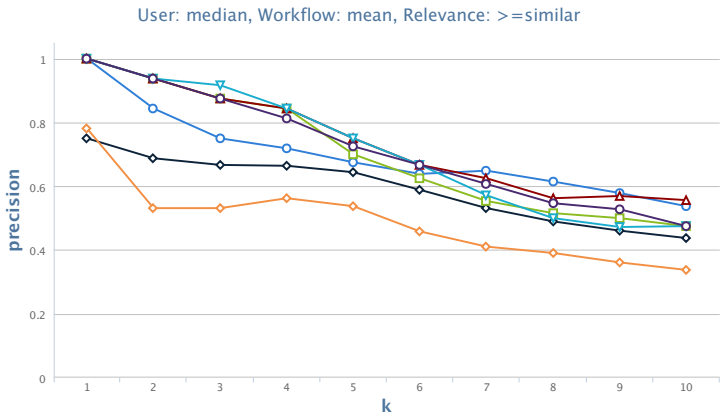


(c)

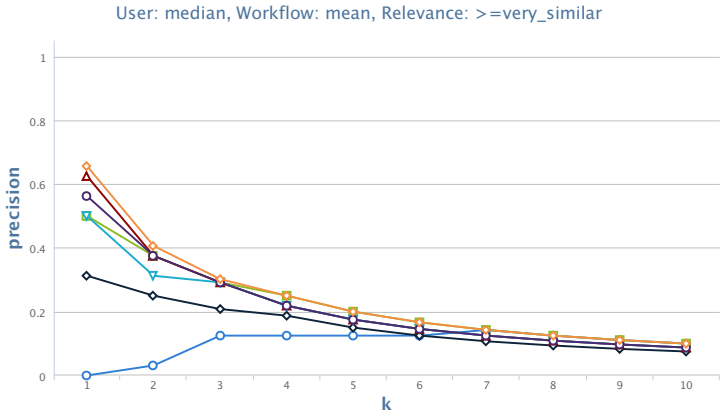
Figure 4.10: Mean retrieval precision at k against the median expert rating for sim_{MS} in various configurations of module similarity assessment (pX), with and without *ip* and *te* for relevance threshold (a) related, (b) similar, and (c) very similar.



(a)



(b)



(c)

Figure 4.11: Mean retrieval precision at k of structural and annotational similarity algorithms for median expert rated relevance of (a) related, (b) similar, and (c) very similar.

of results on the level of whole workflows. The use of external knowledge is taken into account on both levels.

Module Comparison

Figure 4.10 graphs retrieval precision at k for sim_{MS} with various module similarity schemes *pX*, with and without *ip* and *te*. Three different relevance thresholds *related*, *similar*, and *very similar* are considered. Interestingly, the differences in retrieval quality decrease with the increase in relevance level up to the point where all configurations deliver similar performance for retrieval of *very similar* workflows. Apparently, finding the most similar results is independent of the module similarity scheme used. For the retrieval of *related* workflows, on the other hand, where more fine grained assessment of similarity is required, differences between the schemes become visible. The strict label matching approach of *plm* performs worst, confirming our observations from the previous experiment. *pw3* and *pll* tie when not using external knowledge. The inclusion of such knowledge improves performance of all configurations, and puts *pll* ahead of *pw3*, both in terms of mean precision and in terms of standard deviation, which is substantially smaller (not shown).

Workflow Similarity

Figure 4.11 shows retrieval precision for the structural and annotational workflow comparison approaches under evaluation. The structural approaches are used with the *pll* module similarity scheme of edit distance comparison of labels, and have been applied with and without *ip* and *te*. Note that for sim_{GE} we only include results for preprocessed workflow graphs (*ip*). The strict topological comparison applied by sim_{GE} does find the most similar workflows equally well as its structural contenders, but provides much worse results when looking for *similar* or *related* workflows: As sim_{GE} puts a strong emphasis on workflow structure, it also retrieves workflows from other domains than the query workflow, which happen to be similar to the query workflow in terms of their graph structure. sim_{MS} and sim_{PS} provide equivalent result quality and provide best results for both *related* and *similar* workflows. Notice that the difference between configurations with *ip* and *te* and those without is most pronounced for retrieval of *related* workflows, where the inclusion of external knowledge improves both mean precision and stability of the algorithms' performance in terms of standard deviation from the mean. sim_{BW}, while less precise in retrieval than sim_{MS} and sim_{PS} is best at retrieving *related* and *similar* workflows, while it fails to deliver workflows with a median expert rating of *very similar* within the very top of its search results for the set of workflows studied.

4.4.3 Evaluation on Second Data Set

Figure 4.12 shows ranking results on our second dataset of Galaxy workflows, for which we repeated our first experiment on workflow ranking using 8 query workflows. The module comparison schemes used are $gw1$, comparing a selection of attributes

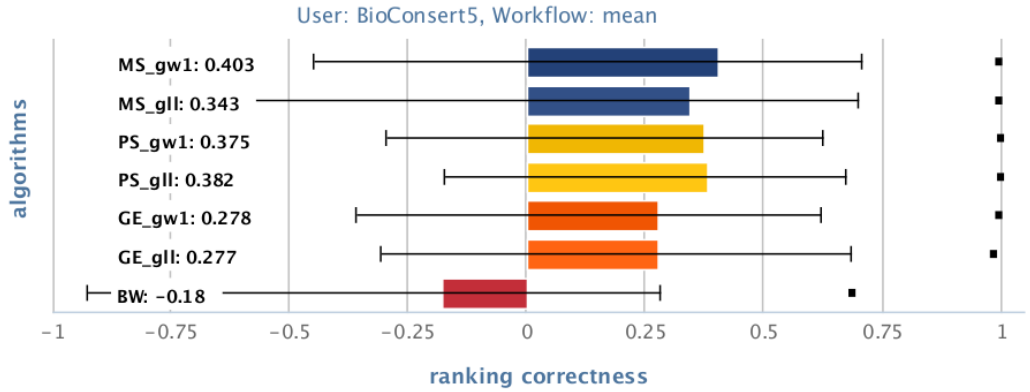


Figure 4.12: Mean ranking correctness for algorithms with different module comparison schemes (see text) on Galaxy workflows.

with uniform weights, and *gll*, comparing only module labels by their edit distance. The most striking observation is that sim_{BW} doesn't provide satisfying results on this data set, as the Galaxy workflows carry less annotations. While, overall, results for the structural algorithms are less convincing than on Taverna workflows, our observations are generally confirmed: structure agnostic comparison by sim_{MS} and comparison respecting substructures by sim_{PS} outperform the strict comparison of full workflow structure performed by sim_{GE}. Interestingly, here label-only comparison of modules offers less correct results than comparison of multiple attributes. As previously observed, sim_{PS} provides more stable results.

4.5 Conclusion

In this study, we compared a multitude of existing approaches to scientific workflow comparison on the to-date largest human-curated corpus of similarity ratings for scientific workflows. We paid special attention to deconstruct every method into their most important conceptual steps and to align these to a common framework, with the goal to increase comparability and to be able to pinpoint observed differences in result quality to their precise cause. Our evaluation clearly showed that each step in the process of workflow comparison makes an important contribution to the overall result quality. While, for practical reasons, our evaluation did focus on workflows from the life sciences domain, the used algorithms are domain agnostic and do not make use of any domain specific knowledge. We do, however, believe that the life sciences are a particularly difficult domain for workflow comparison, due to the large number of different groups developing different tools (and workflows), even for similar tasks, leading to difficult task-matching issues. Our most important findings are:

1. For module comparison, the edit distance of module *labels* seems to be the best approach: It provides best results in retrieval and does not require refinement

of multiple attribute weights; and it provides a more fine grained assessment of similarity than label matching, which, in turn, can only be recommended for retrieval of the few most similar results. Of course, these findings are only valid if labels are telling, i.e., are indicative for the functionality of the labeled module. Such workflows include the studied Taverna workflows from the myExperiment repository, but also the majority of workflows found in the SHIWA repository [3].

2. We have shown that structural approaches can outperform annotational ones when configured appropriately. Especially in repositories where workflows are not well annotated by descriptions or tags, such as the Galaxy repository [45] inspected here, or the CrowdLabs repository of VisTrails workflows [64], structural approaches are indispensible. While full structural comparison by *Graph Edit Distance* appears to be to strict - similar to label matching on the module level -, comparing workflows either by substructures such as paths or by the sets of modules they contain provide comparably convincing results. This is good news, as module set comparison is computationally far less complex than comparing substructures. Yet, the fact that *Path Sets* comparison is more stable in its results across different configurations indicates room for further research to include topological information with less computational complexity.

3. Normalization of the similarity values derived from workflow comparison wrt workflow size is, as clearly shown, indispensable for similarity search of scientific workflows.

4. Next to the intrinsic steps of workflow comparison, we have also looked at several options for further tuning. The use of external knowledge, potentially derived from the workflow repository itself, reduces computational complexity and often improves result quality. Yet, manual acquisition of such knowledge as done in this study, requires extra work to be invested prior to comparisons; furthermore, properties derived from a given repository usually are not transferable to other repositories. Finding suitable ways to automatically derive the required knowledge from the repository, is an interesting area of future research.

5. Another line of future work is to further investigate ensembles of different algorithms: we have shown that such ensembles can significantly improve result quality when compared to single algorithms. The approach of using the algorithms' mean similarity presented here provides a starting ground, leaving room for testing advanced methods such as boosting or stacking [62].

5 Layer Decomposition Similarity of Scientific Workflows

The results of the comprehensive evaluation of existing approaches to scientific workflow comparision presented in the previous chapter indicate that a) structure-based methods are indispensable for some current repositories which lack rich annotations, b) structure-based methods, once properly configured, outperform annotation-based methods even when such rich annotations are available, and c) any such standalone approach is further beaten by ensembles of annotation-based and structure-based methods. We also discovered that both the amount of configuration required and runtime considerations were drawbacks to such methods: Fast workflow comparison using annotations on the workflows' modules provides best results only when ubiquitous, functionally unspecific modules are removed from the workflows in a preprocessing step. The configuration of which modules are to be removed is specific to a given dataset, and is non-trivial. Methods based on workflow substructures, on the other hand, provide rather stable results across different configurations, but have prohibitive runtimes.

In this chapter, we present a novel technique for measuring workflow similarity that accounts for the directed dataflow underlying scientific workflows. The central idea is the derivation of a *Layer Decomposition* for each workflow, which is a compact, ordered representation of its modules, suitable for effective and efficient workflow comparison. We show that the algorithm a) delivers the best results in terms of retrieval quality when used stand-alone, b) is faster than other algorithms that account for the workflows' structure, and c) can be combined with other measures to yield better retrieval at even higher speed. Furthermore, the method is essentially configuration free, which makes it applicable off-the-shelf to any workflow repository.

In the remainder of this chapter, we first review direct contenders from related work on similarity measures for scientific workflows in Section 5.1. In Section 5.2, we briefly summarize those results and settings from the evaluation performed in Chapter 4, which our new algorithm founds on. Section 5.3 introduces our Layer Decomposition algorithm for scientific workflow comparison, which we comparatively evaluate in Section 5.4. We conclude in Section 5.5.

This work appeared in [80].

5.1 Related Work

We here recapitulate a number of prominent approaches to scientific workflow comparison in increasing order of using topological information; a detailed review and comparison of these approaches has been presented in Section 4.1.

Using only workflow annotations, Costa et al. [25] derived *bags of words* from the descriptions of workflows to determine workflow similarity. *Keyword tags* assigned to workflows are explored by Stoyanovich et al. [82]. The *sets of modules* contained in workflows have been used by Silva et al. [78], Santos et al. [76], and Friesen et al. [39]. Bergman et al. [8] extended this idea by additionally comparing the workflows' datalinks based on semantic annotations; however, such semantic annotations are not generally available in publicly available workflow repositories. Considering workflow substructure, the use of *Maximum Common Isomorphic Subgraphs* (MCS) has been investigated by Goderis et al. [43], Santos et al. [76], and Friesen et al. [39]: the latter also proposed using *Graph Kernels* derived from frequent subgraphs in a repository. Subgraph comparison has also been applied in similar domains, e.g., by Corrales et al. [24] to compare BPEL workflows or by Krinke [54] to compare program dependence graphs. Finally, using the full graph structure, Xiang et al. [88] compute scientific workflow similarity from their *Graph Edit Distance*.

5.2 Preliminaries

In the previous chapter, we reported on a comprehensive evaluation of previous approaches to workflow similarity search. In addition to comparing retrieval quality quantitatively, we also introduced a framework for qualitatively comparing different systems (the relevant part of which is shown in Figure 5.1). This is an important tool, as the process of workflow comparison entails many steps, of which a concrete topology-comparison algorithm is just one. First, the similarity of each pair of modules from two workflows is determined using pairwise *module comparison*. Second, using these pairwise module similarities, a *mapping of modules* onto each other is established. This mapping may be influenced by the *topological decomposition* of the workflows imposed by the third step of *topological comparison*, which in turn uses the established mapping to assess the similarity of the two workflows. Finally, *normalization* of the derived similarity value wrt the sizes of the compared workflows may be desirable. This process of scientific workflow comparison is preceded by an (optional) additional preprocessing step. Such preprocessing may, for instance, alter the workflows' structure based on externally supplied knowledge about the elements it contains (see Section 5.2.4).

Each of these steps has a notable impact on the concrete values computed and thus the assessment of different algorithms. While we evaluated a number of possible options for each step in the previous chapter, we here recall the settings we will focus evaluation of our novel algorithm on. Note that our novel Layer Decomposition

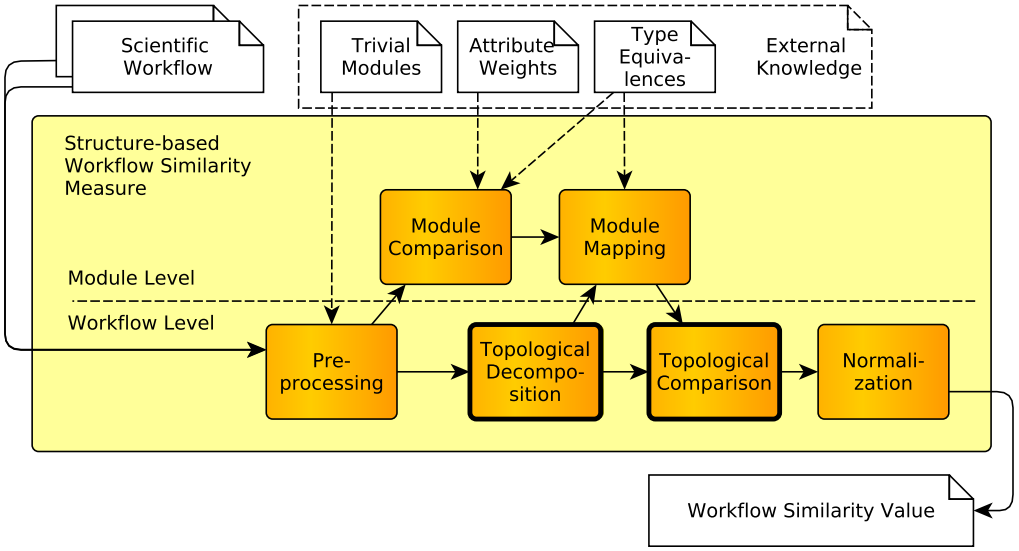


Figure 5.1: Workflow comparison process, see also Figure 4.2

method (described in the next section) is a contribution dedicated to the step of *topological decomposition and comparison*; in Section 5.4, we will compare different approaches to this step while keeping all other steps constant.

5.2.1 Module Comparison

Structure-based approaches to workflow similarity perform some type of comparison of the two workflow graphs. To this end, they must be able to measure the similarity of two nodes in the graphs, which represent two modules in the to-be-compared workflows. Module comparison is typically approached by comparing the values of their attributes. These range from identifiers for the *type* of operation to be carried out, to the descriptive *label* of the module given to it by the workflow's author, to rather specific attributes such as the *url of a web-service* to be invoked. In Chapter 4 we compared several combinations of attributes with different weightings and showed that choosing a suitable configuration is most crucial for result quality of structural workflow comparison. Here, we will use the following three schemes to test the impact of module comparison on the topological comparison provided by our new algorithm:

- *pw0* assigns uniform weights to all attributes and compares module *type*, and the web-service related properties *authority name*, *service name*, and *service uri* using exact string matching. Module *labels*, *descriptions*, and *scripts* in scripted modules are compared using Levenshtein edit distance.
- *pw3* compares single attributes in the same way as *pw0* but uses higher weights for *labels*, *script* and *service uri*, followed by *service name* and *service authority*

in the order listed. This weighting resembles the proposal of [78].

- *pll* disregards all attributes but the *labels* and compares them using the Levenshtein edit distance, which is the approach taken in [8].

5.2.2 Module Mapping

Structural comparison of two graphs may be carried out by comparing all nodes of one graph to all nodes of the other graph, or by first computing a *node mapping* which defines the set of allowed node associations (often one-to-one). Only then do graph operations such as node deletion or node insertion make sense. We here use two strategies to obtain such a mapping, depending on the amount of topological information available:

- *maximum weight matching (mw)* chooses the set of one-to-one mappings that maximizes the sum of similarity values for unordered sets of nodes.

- *maximum weight non-crossing matching (mwnc)* [63] requires an order on each of the two sets to be mapped to be given by the graphs' topological decompositions. Given two ordered lists of modules $(m_1, ..m_i, ..m_k)$ and $(m'_1, ..m'_j, ..m'_l)$, a mapping of maximum weight between the sets is computed with the additional constraints no pair of mappings (m_i, m'_j) and (m_{i+x}, m'_{j-y}) may exist with $x, y \geq 1$.

5.2.3 Normalization

A difficult problem in topological comparison of workflows is how to deal with differences in workflow size. In particular, given a pair V, W of workflows where $V \subset W$ and $|W| >> |V|$, what is their similarity? This decision typically is encoded in a normalization of similarity values wrt the sizes of the two workflows [79]. In this work, we use a variation of the Jaccard similarity coefficient, which measures the similarity of two sets A and B by their relative overlap: $\frac{|A \cap B|}{|A|+|B|-|A \cap B|}$. We modify this formula because the methods for comparing modules do not create binary decisions but instead return a similarity score; details can be found in Section 4.1.

5.2.4 External Knowledge

When comparing two workflows, knowledge derived from the entire workflow repository or even from external sources may be taken into account. While this idea in theory could be implemented by complex inferencing processes over process ontologies and formal semantic annotations, our results in Chapter 4 showed that the following two simple and efficiently computable options are already quite effective:

- *Importance Projection Preprocessing.* Many modules in real-world workflows actually convey little information about workflow function, but only provide parameter settings, perform simple format conversions, or unnest structured

data objects [81]. Importance projection is the process of removing such modules from a workflow prior to its comparison, where the connectivity of the graph structure is retained by transitive reduction of removed paths between the remaining modules. Note that this method requires external knowledge given in the form of a method to assess the contribution of a given module to the workflow's function, which is a rather strong requirement. The implementation provided here relies on manual assignments of importance based on the type of operation carried out by a given module.

- *Module Pair Preselection.* Instead of computing all pairwise module similarities for two workflows prior to further topological comparisons, this method first classifies modules by their type and then compares modules within the same class. This reduces the number of (costly) module comparisons and may even improve mapping quality due to the removal of false mappings across types. Here, external knowledge must be given in the form of a method assigning a predefined class to each module.

5.3 Layer Decomposition Workflow Similarity

In this section, we present a novel approach, called *Layer Decomposition* (LD), for the structurally comparing two workflows. The fundamental idea behind LD is to focus on the order in which modules are executed in both workflows by only permitting mappings of modules to be used for similarity assessment which respect this order (in a sense to be explained below). Two observations led us to consider execution order as a fundamental ingredient to workflow similarity. First, it is intuitive: The function of a workflow obviously critically depends on the execution order of its tasks as determined by the direction of data links; even two workflows consisting of exactly the same modules might compute very different things if these modules are executed in a different order. Nevertheless, most structural comparison methods downplay execution order. For instance, it is completely lost when only module sets are compared, and a few graph edits can lead to workflows with very different execution orders (like swapping the first and last of a long sequence of modules). Second, we observed in our previous evaluation (Chapter 3) that approaches to topological workflow comparison which put some focus on execution order are much more stable across different configurations of the remaining steps of the workflow comparison process. In particular, comparing two graphs using their path sets, i.e., the set of all paths from a source to a sink, produced remarkably stable results both with and without the use of external knowledge. Inclusion of such knowledge in workflow comparison had among the largest impact on the overall performance of methods, but requires corpus-specific expert intervention. Based on these findings, developing methods that achieve retrieval results of high quality without requiring external knowledge seemed like a promising next step.

In the following, we first explain how LD extracts an ordering of workflow modules from the workflow DAG. We then show in Section 5.3.2 how two workflows can be

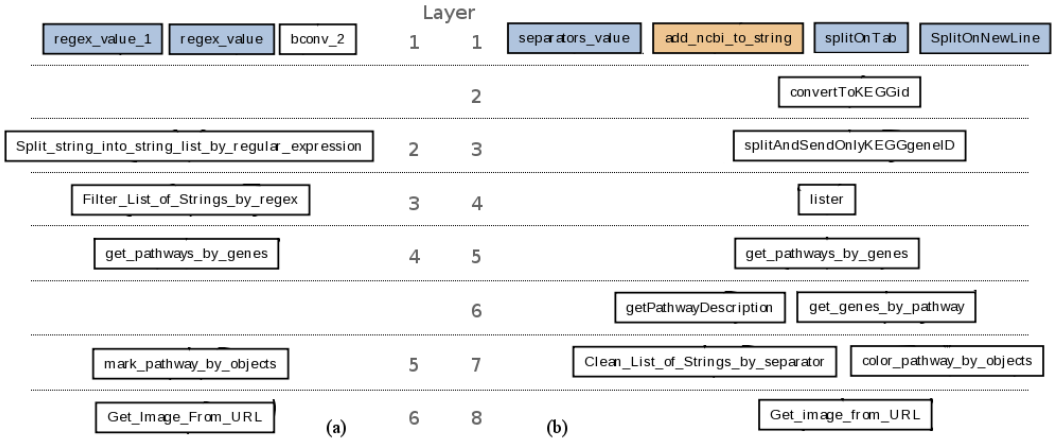


Figure 5.2: Sample layer decomposition and layer mapping of scientific workflows (a) 1189, (b) 2805; see also Fig. 4.1).

effectively compared using this partial ordering. Finally, we explain in Section 5.3.3 how normalization is performed.

5.3.1 Topological Decomposition

The *linearization* (or *topological sort*) of a DAG is an ordering of its nodes V such that node u precedes node v in the ordering, if an edge (u, v) exists. Obviously, a DAGs linearization can be computed in linear time using topological sorting; however, it is generally not unique. As the quality of the subsequent mapping (see below) depends on the concrete linearizations chosen for the two workflows under consideration, it is important to find a good pair of linearizations, i.e., linearizations such that highly similar modules will later get mapped onto each other. Since the number of possible linearizations is $\Omega(n!)$ (where n is the number of modules in a workflow), assessing all possible pairs is generally infeasible; it is also infeasible in practice, as many real life workflows have many different linearizations (for instance, 23.5% of the 1485 Taverna workflows in our evaluation set have more than 100 different linearizations).

We tackle this problem by representing all possible linearizations of a given workflow in a concise data structure. Observe that a DAG has more than one linearization iff between two consecutive nodes in one of its linearizations no direct datalink exists, because in this case swapping the two nodes creates another linearization. In all such cases, we tie the two nodes in question into a single position in the ordering. We call such a tie at position i a *layer* L_i. Compacting all sequences of two or more swappable nodes of a linearization in this way yields a layered ordering of the DAG which we call its *layer decomposition* $LD = (L_1, .., L_i, .., L_k)$. Note that the layer decomposition of a DAG is unique, as a) layers themselves are orderless sets of modules, and b) following from the definition of the underlying *linearization*,

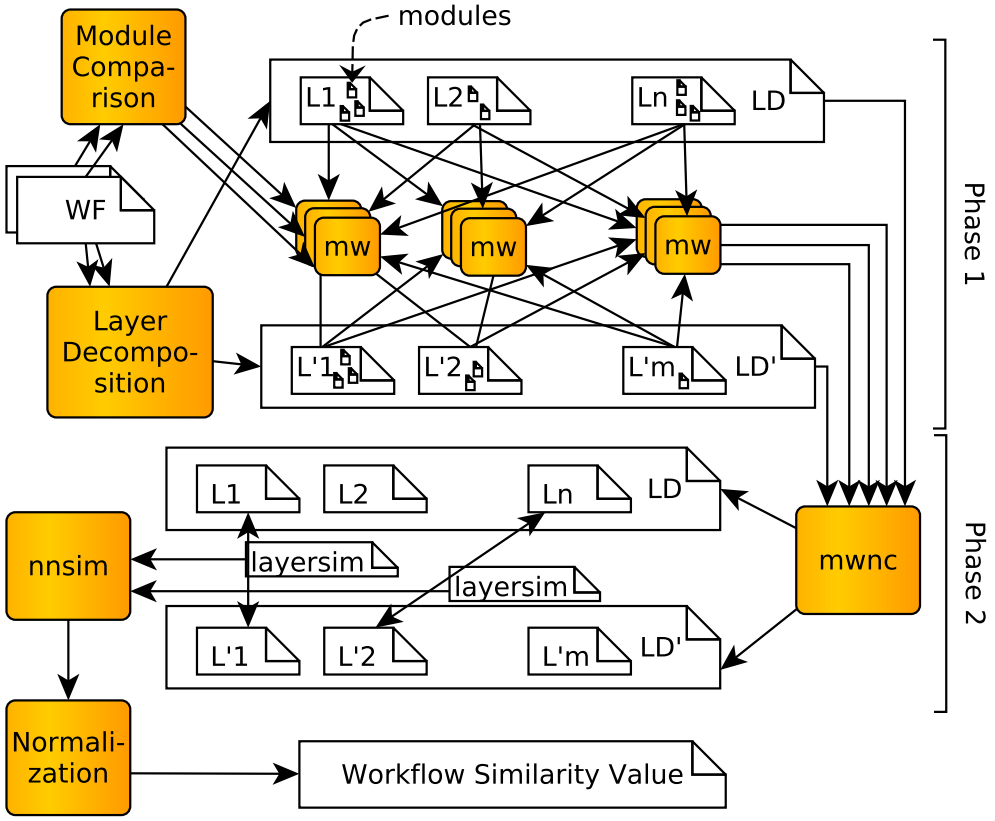


Figure 5.3: Overview of workflow comparison applied by LD

for any L_i and L_j such that $i < j$, for every $v \in L_j$, there is some $u \in L_i$ which precedes v in every linearization, which c) uniquely defines the positions of L_i and L_j within the decomposition. To compute a workflow's layer decomposition, we use a simple iterative algorithm. First, all modules with in-degree 0 (the DAGs source nodes) form the top layer L_1. These modules and all their outbound data links are removed from the workflow; this process is repeated until no more modules remain. Figure 5.2 shows the layer decompositions of the sample workflows introduced in Figure 4.1 on page 50. In Figure 5.2, the different layers are visually aligned to reflect their mapping, as it is derived in the following step.

5.3.2 Topological Comparison

The layer decomposition of a workflow partitions its module set by execution order creating an ordered list of module subsets. To compare the layer decompositions LD and LD' of two workflows wf and wf', respectively, we take a two-phase ap-

proach, sketched in Figure 5.3. First, pairwise similarity scores for each pair of layers $(L, L') \in LD \times LD'$ are computed from the modules they contain using the maximum weight matching (mw), based on the similarity values $p(m, m')$ derived by a given module comparison scheme as introduced in Section 5.2.2:

$$layersim(L, L') = \sum p(m, m') \mid (m, m') \in mw(L, L')$$

In the second phase, the ordering of the layers - and thus of the modules they are comprised of - is exploited to compute the decompositions' maximum weight non-crossing matching (mwnc) with the pairwise similarities of layers from phase one. The resulting layer-mapping serves as the basis for the overall (yet non-normalized) similarity score of the compared workflows using LD:

$nnsim_{LD}(wf, wf') =$

$$\sum layersim(L, L') \mid (L, L') \in mwnc(LD, LD')$$

5.3.3 Normalization

As done for all other methods we shall compare to, we normalize the similarity values computed by LD using the Jaccard variation described in Section 5.2.3. Thus, the final, normalized LD-similarity is computed as:

$$sim_{LD}(wf, wf') = \frac{nnsim_{LD}}{|LD| + |LD'| - nnsim_{LD}}.$$

We analogously normalize $layersim(L, L')$ by $|L|$ and $|L'|$. This way, if two workflows are identical, each layer has a mapping with a similarity value of 1. Then $nnsim_{LD} = |mwnc(LD, LD')| = |LD| = |LD'|$, and $sim_{LD} = 1$.

5.4 Evaluation

We evaluate our novel Layer Decomposition algorithm on the gold standard corpus of workflow similarity ratings given by workflow experts, which we assembled in Chapter 4. The corpus contains 2424 similarity ratings from 15 experts from four countries for 485 workflow pairs from a set of 1485 Taverna workflows, given along a four step Likert scale [58] with the options *very similar*, *similar*, *related*, and *dissimilar* plus an additional option *unsure*. The ratings are grouped by 24 *query workflows*. Each query workflow has a list of 10 workflows compared to it, which are ranked by a consensus computed from the experts rankings using BioConsert Median Ranking [20]. These rankings are used to evaluate the algorithms performance in *workflow ranking*. For 8 of the query workflows, additional expert ratings are available covering all workflows which were ranked among the top-10 most similar workflows by a selection of algorithms to be further evaluated, when run against

Table 5.1: Algorithm shorthand notation overview

	Notation	Description
Algorithms	LD	*Layer Decomposition* topological comparison
	MS	*Module Sets* topological comparison
	PS	*Path Sets* topological comparison
	GE	*Graph Edit Distance* topological comparison
	BW	*Bag of Words* annotation based comparison
	BT	*Bag of Tags* annotation based comparison
Configurations	np	No structural preprocessing of workflows
	ip	*Importance projection* workflow preprocessing
	ta	No module pair preselection for comparison
	te	*Type equivalence* based module pair preselection
	pw0	Module comparison with uniform attribute weights
	pw3	Module comparison on tuned attribute weights
	pll	Module comparison by edit distance of labels only

the entire workflow corpus. These are used to assess the performance in *workflow retrieval*.

Using this corpus, we compare the LD algorithm against approaches based on Module Sets, Path Sets, Graph Edit Distance, Bags of Words, and Bags of Tags (as presented in Section 5.1). First, we investigate the algorithms' performance in the tasks of workflow ranking (Section 5.4.1) and workflow retrieval (5.4.2). Second, we compare the runtimes of the different topological comparison methods (5.4.3). Third, we evaluate whether the successive application of multiple algorithms in retrieval can improve result quality and its implications on runtime (5.4.4). Finally, in Section 5.4.5 we shall confirm the previous results on ranking performance using a second data set, from the Galaxy repository.

5.4.1 Workflow Ranking

To evaluate ranking performance, we use the measures of ranking *correctness* and *completeness* [8, 19]. For ranking correctness, the order of each pair of elements in the experts' consensus ranking and the algorithmic ranking is compared, counting pairs sharing the same order and pairs that don't, to determine the correlation of the compared rankings. Values range from -1 to 1, where 1 indicates full correlation of the rankings, 0 indicates that there is no correlation, and negative values are given to negatively correlated rankings. Ranking completeness, on the other hand, measures the number of pairs of ranked elements that are not tied in the expert ranking, but tied in the evaluated algorithmic ranking. The objective here is to penalize the tying of elements by the algorithm when the user distinguishes their ranking position.

Figure 5.4 shows ranking performance for sim_{LD} in direct comparison to sim_{MS},

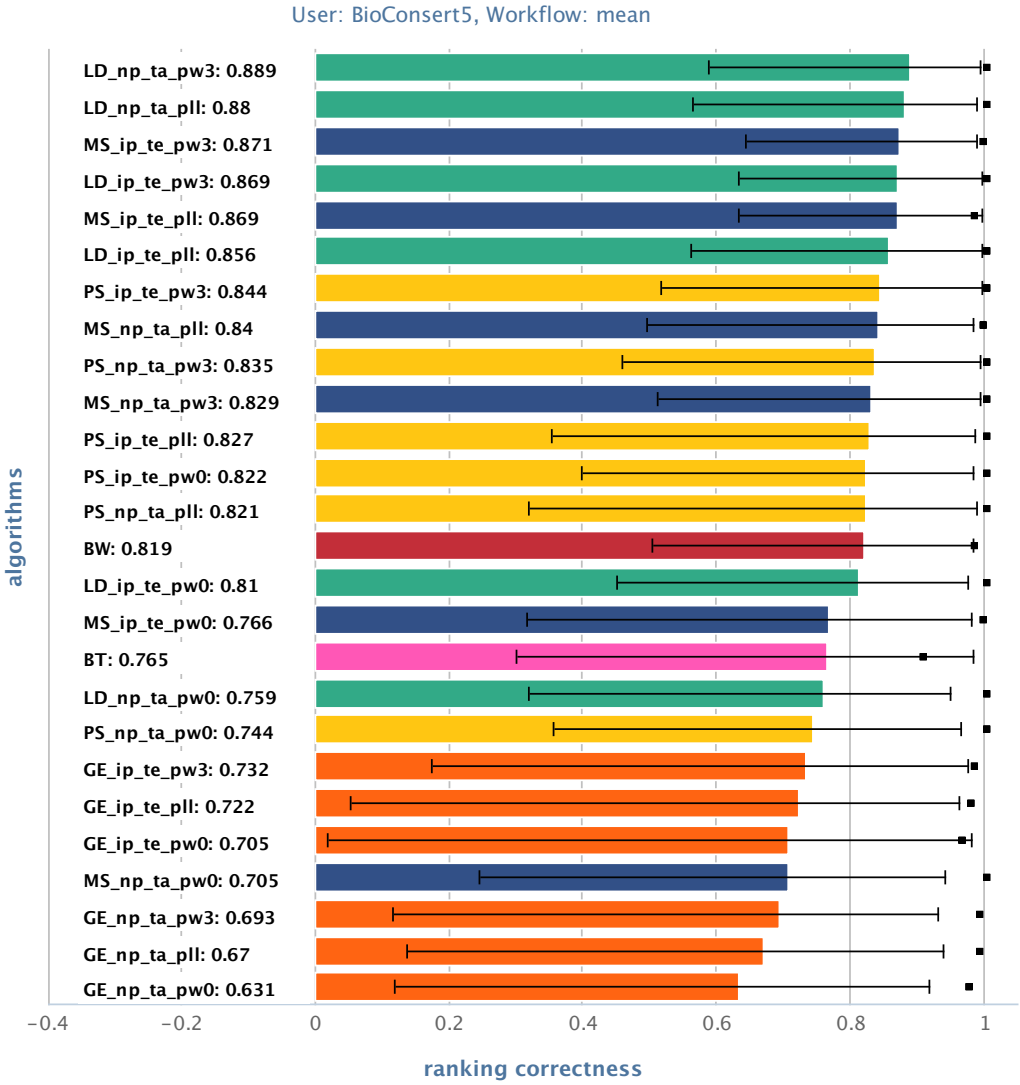


Figure 5.4: Mean ranking correctness (bars) with upper and lower stddev (error-bars), and mean ranking completeness (black squares) over 24 lists of 10 workflows for different algorithms and configurations (see Table 5.1 for notation).

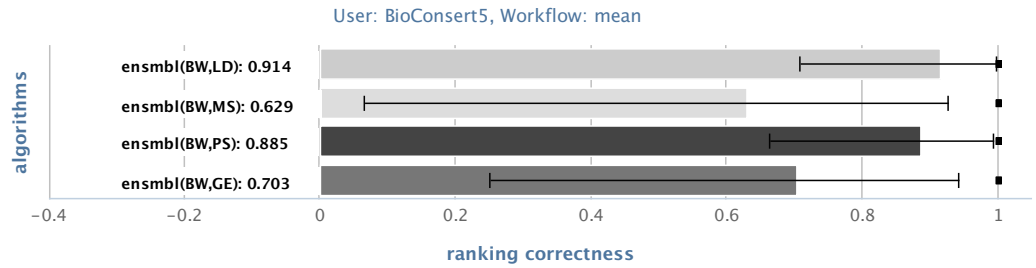
sim_{PS}, sim_{GE}, and the annotation based measures sim_{BW} and sim_{BT}. Results are sorted by mean ranking correctness. Each algorithm is applied in a variety of different configurations (see Chapter III for options; see Table 5.1 for notation). Regarding ranking correctness, several observations can be made: Firstly, sim_{LD} provides best results. Secondly, both sim_{LD} and sim_{PS} provide most stable results across different configurations. Performance of sim_{MS}, on the other hand, varies with the quality of the module comparison scheme used, and especially with the use of external knowledge in terms of *ip*. Thirdly, while sim_{MS}, when configured properly, can achieve ranking correctness values comparable to the best results of sim_{LD}, sim_{PS} is generally slightly behind sim_{LD}. In contrast, sim_{GE}, putting a high emphasis on overall workflow structure, does not provide competitive results; we therefore omit it in all further evaluations. As for ranking completeness, we see that both sim_{LD} and sim_{PS} fully distinguish all workflows in terms of their similarity to the query workflows where users make a distinction as well. The ranking provided by sim_{MS}, on the other hand, is often only near-complete.

Figure 5.5 shows the results of selected ensembles of different base methods using mean similarity values. Generally, result quality improves considerably. For instance, the top performing standalone configuration of sim_{LD} achieves a ranking correctness of 0.889, and is not only outperformed by its combination with sim_{BW} by 2.5 %-points, but especially in terms of stability of results across different query workflows, as apparent from the standard deviations from the mean. The ensembles including sim_{LD} deliver best results, outperforming other ensembles especially when no external knowledge is used.

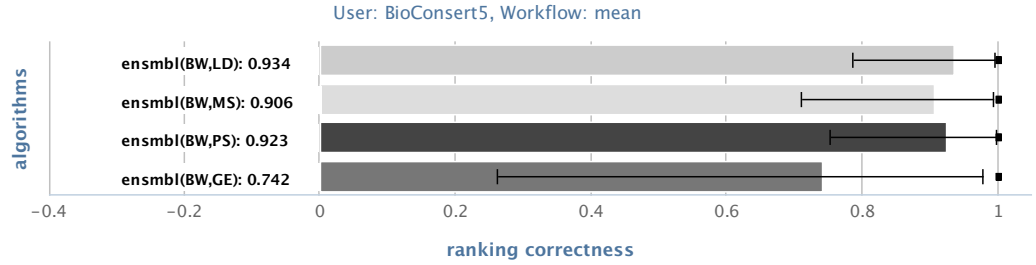
5.4.2 Workflow Retrieval

Figure 5.6 shows precision at k [65] for each position in the top-10 results returned by each of the algorithms. We focus our presentation on configurations using the *pll* module comparison scheme. Comparing sim_{MS}, sim_{LD}, and sim_{PS}, Figure 5.6a shows that when treating results as relevant with a median expert rating of at least *related*, all algorithms deliver results of similar (very high) quality. The usage of *ip* and *te* improves results for all algorithms. While this may seem a contradiction to the more distinguished results of the *workflow ranking* experiment at first, it has to be kept in mind that differences in ratings between the results retrieved are not considered for evaluation of retrieval precision. This becomes more clear when inspecting algorithmic retrieval performance for a relevance threshold of *similar* (Fig. 5.6b). Here, the improved ranking results of sim_{LD} are reflected both by a slight advantage in the very top results returned, especially when *ip* is used, and by a more pronounced improvement of mean retrieval performance for the second half of the top-10 results.

Detailed inspection of the workflows retrieved by each of the algorithms revealed that the nature of sim_{LD}'s topological comparision favours retrieval of perfectly matching substructures over global workflow matches with slightly reduced pairwise layer similarities, due to the way the structural complexity of the workflows



(a)



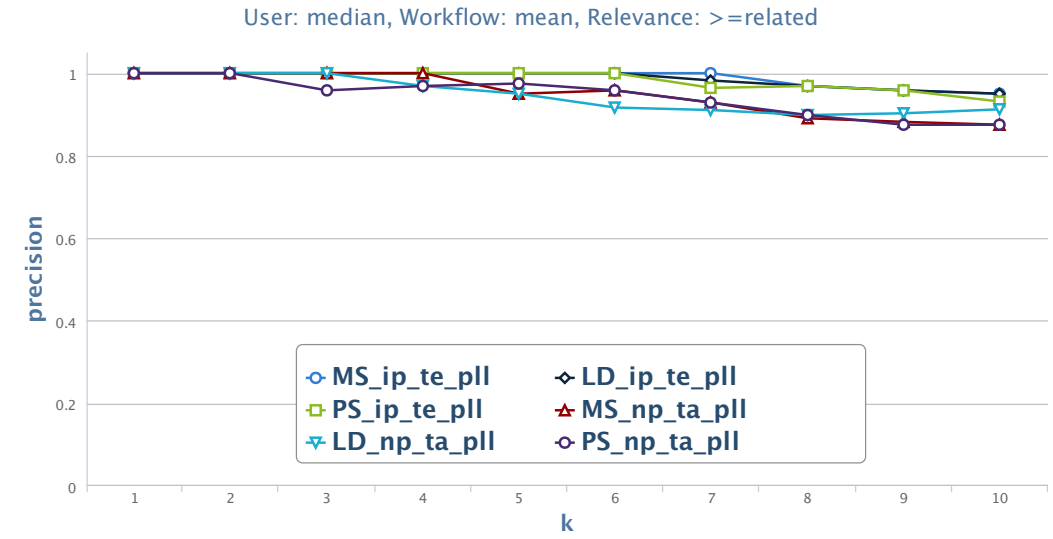
(b)

Figure 5.5: Mean ranking correctness for ensembles of Bag of Words workflow comparision and each structural algorithms (a) without and (b) with *ip* and *te*, each using *pll* module comparison.

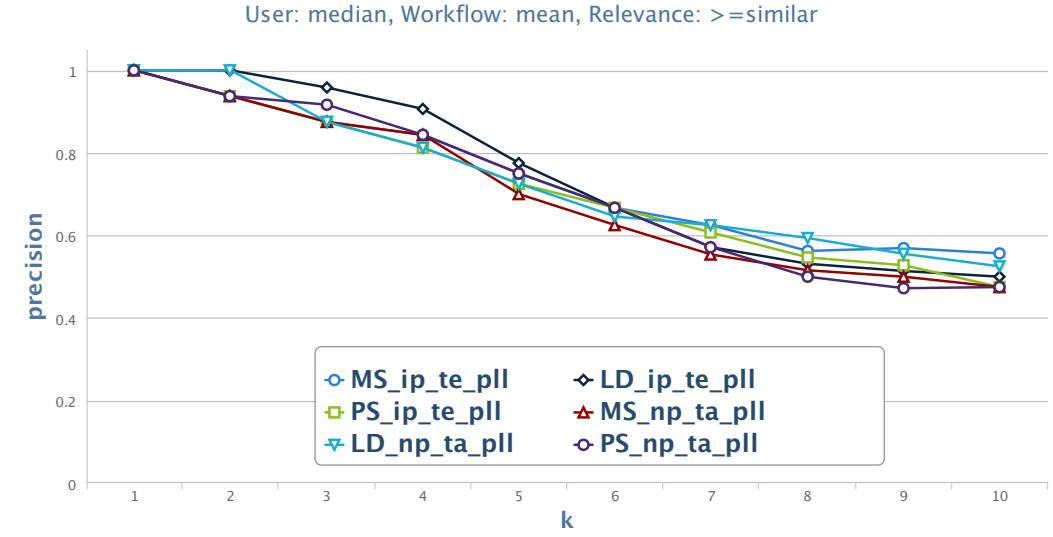
is reduced to a more 'fuzzy' representation in the layers. This behaviour results in some false positive retrievals for a small fraction of the query workflows used. To account for this, we extended the original algorithm by adding a penalty for layers not matched in the maximum weight non crossing matching, when the number of such mismatched layers exceeds a configurable percentage of the layers in the larger of the compared workflows (i.e., the maximum number of layers that could possibly be matched). Setting this allowance to 25% notably improves retrieval performance of the sim_{LD} algorithm, as shown in Figure 5.7. Yet, selection of the right mismatch allowance to be used does depend on the concrete dataset, and, as such, requires prior, in depth knowledge about the repository to be queried. We thus refrain from applying this extension in the remainder of this evaluation.

5.4.3 Runtime

It can be expected that the topological comparison performed by the Layer Decomposition algorithm entails a penalty in runtime when compared to the topology-agnostic Module Set approach. Here, we investigate how big this penalty is, how it compares to other algorithms, and how much it depends on the sizes of the compared workflows by measuring runtimes for sim_{LD}, sim_{MS}, and sim_{PS}. For runtime measurement, each of the 1485 workflows in our dataset was compared against itself by each of the algorithms to obtain a unbiased sample wrt typical workflow sizes.

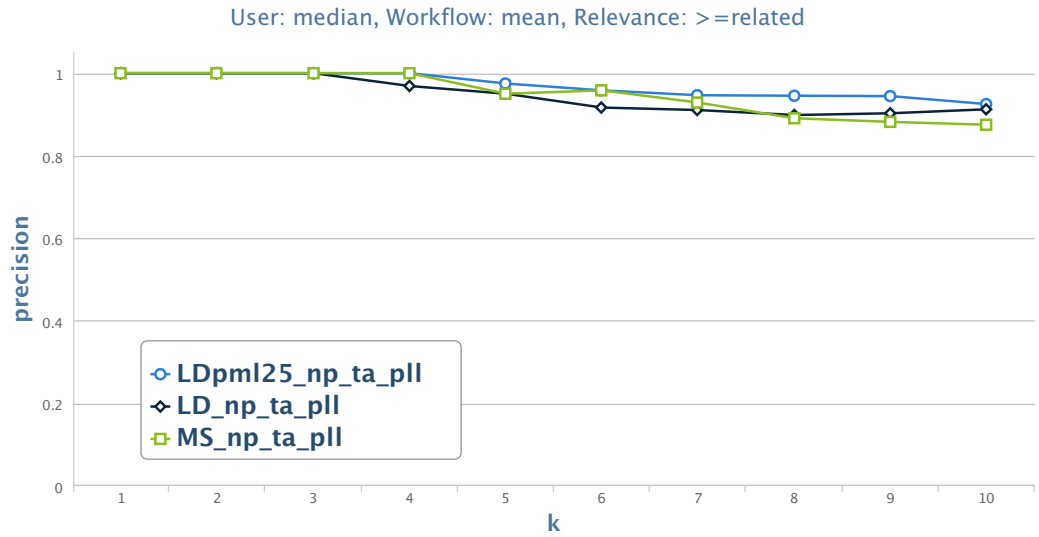


(a)

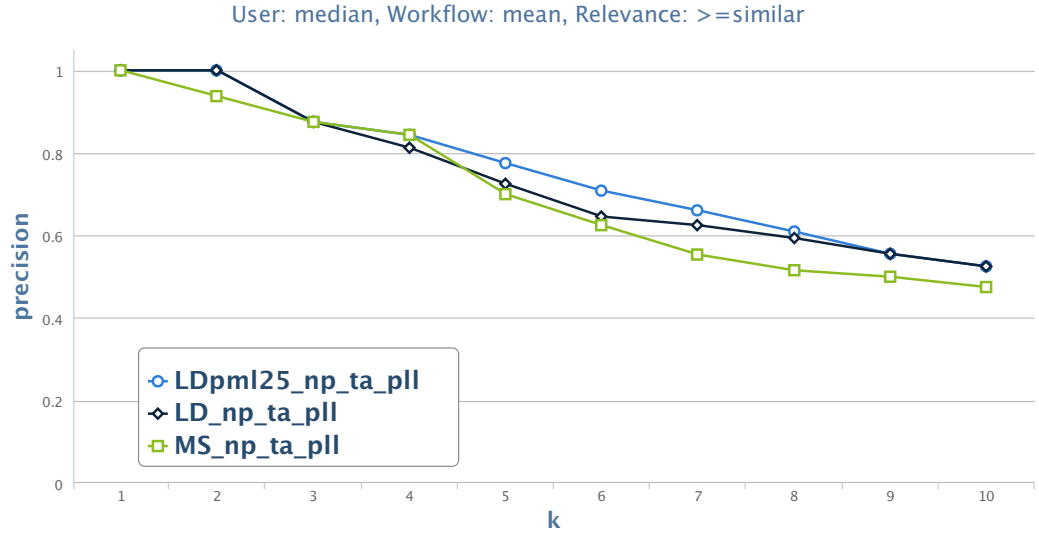


(b)

Figure 5.6: Mean retrieval precision at k against the median expert rating for structural similarity algorithms for relevance threshold (a) related, and (b) similar. Algorithms used with module similarity by edit distance of labels (*pll*), with and without *ip* and *te*.



(a)



(b)

Figure 5.7: Mean retrieval precision at k for similarity algorithms MS and LD and refined LD penalizing mismatched layers exceeding 25% of the larger workflows layers (LDpml25) for relevance thresholds by median expert rating of (a) related, and (b) similar.

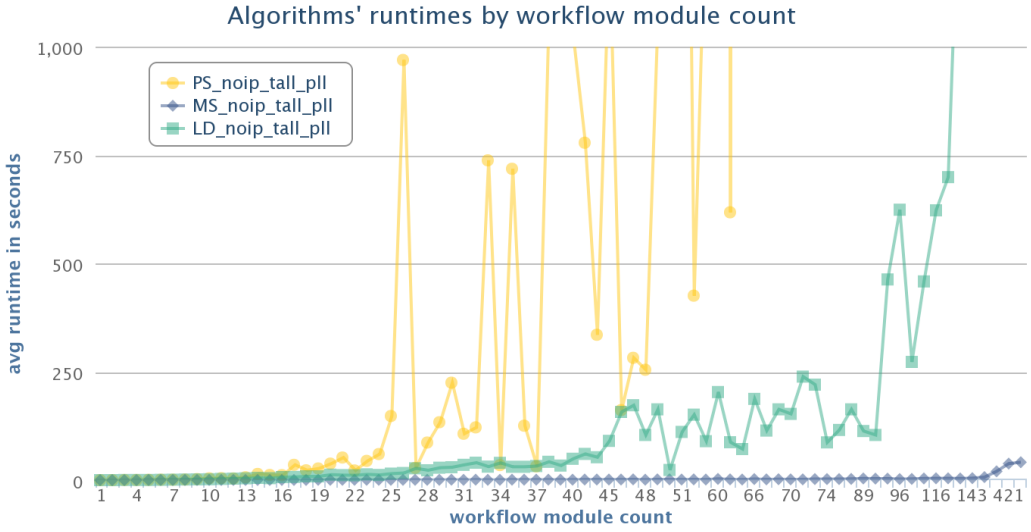


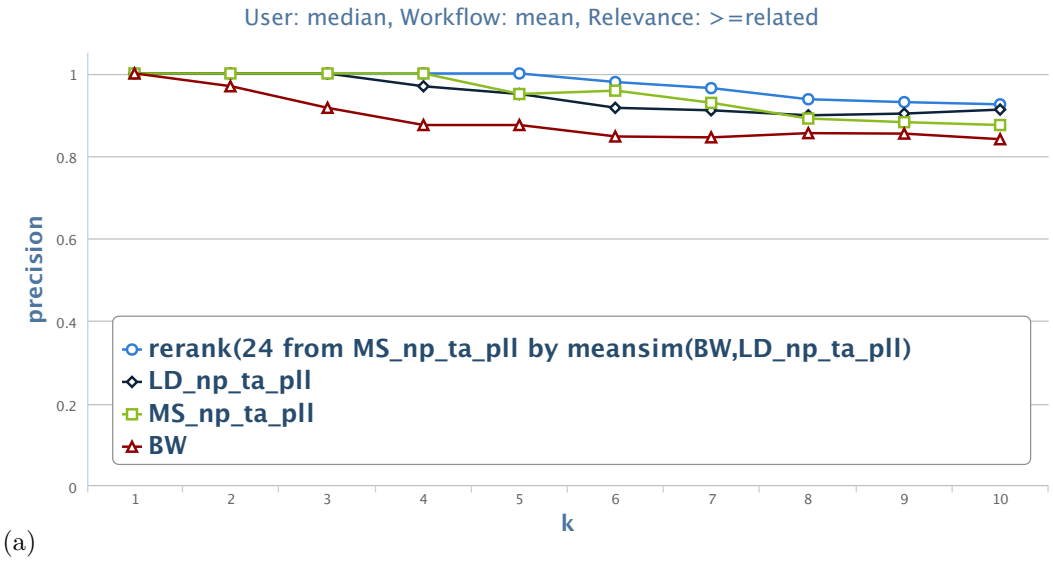
Figure 5.8: Algorithm runtimes by workflow size.

Each comparison was done 5 times and results were averaged.

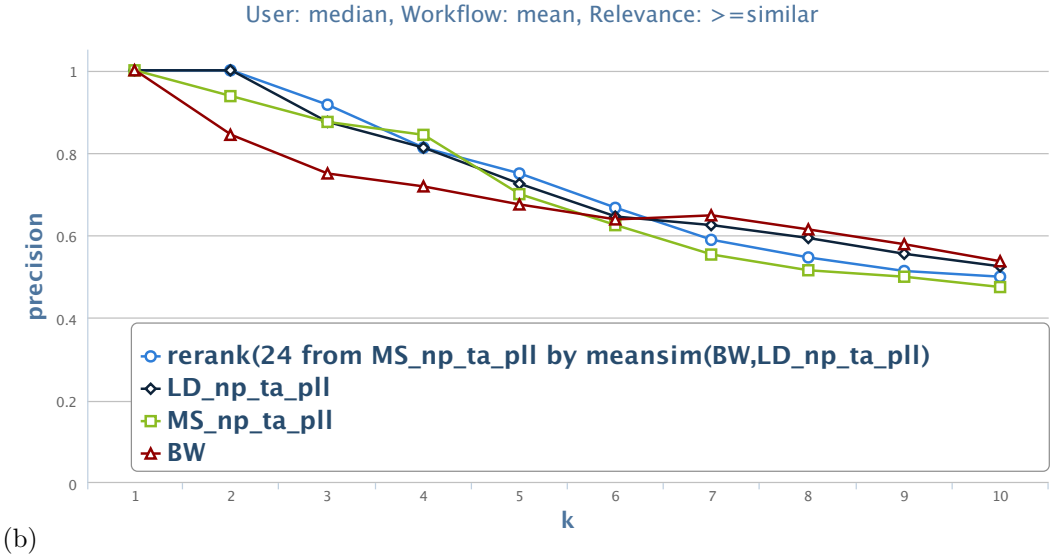
Figure 5.8 shows average runtimes, grouped by workflow sizes ranging from 1 to 437 modules. The average number of modules per workflow in our dataset is 11.4 (see also Chapter 3 and [87]). Note that the figure only shows the time taken for the actual topological comparison. Steps that can be performed offline or only have to be performed once per query execution, such as decomposition of the workflows into the sets of paths or layers, are not considered; module similarities have been precomputed and cached. While runtimes of all algorithms are comparably low for workflow sizes up to around 15 modules, clearly, the only algorithm with acceptable runtimes also for larger workflows is sim_{MS}. sim_{PS} runtimes vary greatly, as these are dominated by the number of different paths the compared workflows contain. This variance is reduced for sim_{LD}, which only needs to compare one pair of decompositions per workflow, resulting in a substantial speedup. Yet, with increasing workflow size and increasing numbers of multi-module layers, runtimes are much higher than with simple Module Set comparison.

5.4.4 Reranked Retrieval Results

Given the findings regarding the deterred runtime of sim_{LD} and the fact that external knowledge to be used in workflow comparison such as *ip* and *te* incurs a severe data acquisition bottleneck, we speculated whether the results of the (fast) Module Set comparison algorithm can be improved - without external knowledge - by merging them with the high-quality ranking performance of sim_{LD}. We therefore performed an experiment where we reranked the top retrieval results of sim_{MS} by the ensemble of sim_{BW} and sim_{LD}. Figure 5.9 shows retrieval precision for



(a)



(b)

Figure 5.9: Mean retrieval precision at k against the median expert rating for similarity algorithms MS, LD and BW, and the top 24 results of MS reranked by the ensemble of BW and LD, for relevance threshold (a) related, and (b) similar.

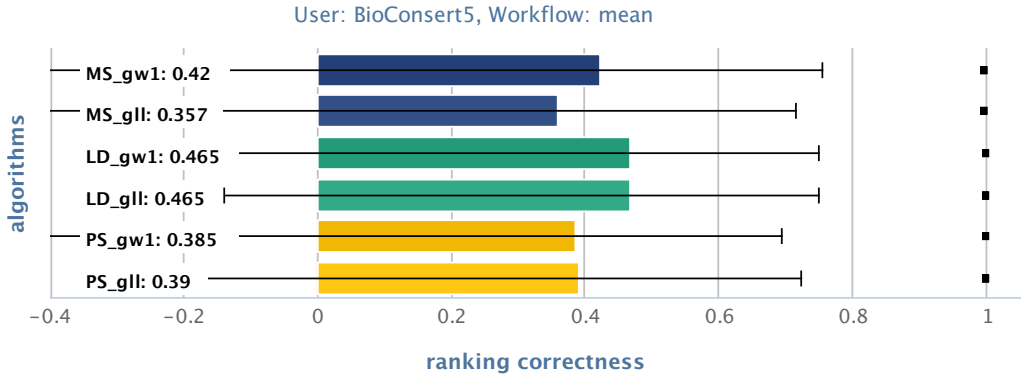


Figure 5.10: Mean ranking results on Galaxy workflows (see text).

sim_{MS}, sim_{LD}, and sim_{BW} on their own, and for the reranked top 24 search results of sim_{MS}. Especially for the relevance threshold of *related*, reranking the results clearly improves performance and makes it comparable to that of algorithm configurations including external knowledge. For a threshold of *similar*, the benefit is less pronounced, yet still observable. We believe that studying in more detail such reranking methods, especially focussing on the trade-off between runtime and result quality, are a prospective venue for further research.

5.4.5 Applicability to Other Datasets

Our gold standard corpus also includes a second set of workflows from another workflow repository, namely the public Galaxy workflow repository. For 8 query workflows, rated lists of compared workflows are available to evaluate ranking performance. This dataset differs from the previous one in various respects: Galaxy workflows are exclusive to the Bioinformatics area, the repository is smaller and curated by a smaller group of people, the annotation is generally more sparse (no tags etc.), and the modules used are only local executables (no web services as frequently in used in Taverna). Looking at such diverse data sets is important to show robustness of any evaluation results.

Figure 5.10 shows ranking correctness for sim_{MS}, sim_{LD}, and sim_{PS} on this second dataset. The module comparison schemes used are $gw1$, comparing a selection of attributes with uniform weights, and gll, comparing only module labels by their edit distance. While results are generally less good than on the myExperiment data set, sim_{LD} here even more clearly outperforms the other algorithms. We are currently looking to extend this dataset to be able to perform a more complete evaluation and to trace back the observed differences in ranking performance to properties of the data set.

5.5 Conclusion

We introduced *Layer Decompositon* (LD), a novel approach for workflow comparison specifically tailored to measuring the similarity of scientific workflows. We comparatively evaluated this algorithm against a set of state-of-the art contenders in terms of workflow ranking and retrieval. We showed that LD provides the best results in both tasks, and that it does so across a variety of different configurations - even those not requiring extensive external knowledge. Results in ranking could be confirmed using a second data set. Considering runtime, we not only showed our algorithm to be faster than other structure-aware approaches, but demonstrated how different algorithms can be combined to reduce the overall runtime while achieving comparable, or even improved, result quality.

Though we did consider runtimes, our evaluation clearly focusses on the quality of ranking and retrieval. Real time similarity search at repository-scale will require further efforts in terms of properly indexing workflows. Such indexing of workflows is straightforward when considering only their modules (like in sim_{MS}), but requires more sophisticated methods when also topology should be indexed. Therefore, our approach of stacking *Layer Decomposition*-based ranking onto workflow retrieval by modules provides a good starting place for applying structure-based workflow similarity to scientific workflow discovery to scale.

6 Accelerating Similarity Search in Scientific Workflow Repositories

In the previous chapters, we have shown that structure-based similarity search for scientific workflows can substantially outperform annotation-based approaches with respect to result quality. A drawback to such structure-based methods is that they are comparatively slow to compute. This consideration of speed becomes important when translating our previous results into a real-world system for similarity search over whole repositories: To be accepted by users, search results have to be presented fast, if not near instantly - making indexing a non-optional requirement. Yet, structure-aware indexing of workflows is not straightforward. For instance, the system proposed in [43] uses subgraph matching for similarity search in a scientific workflow repository. Their use of an existing graph indexing library requires significant workarounds for the intended purpose which cause a substantial slowdown of the resulting system. Not resorting to such existing libraries, [9] introduce a system for workflow similarity search using a two-phase retrieval: After an initial, rough preselection of (potentially) suitable workflows from a fast search over the whole repository, only some candidate workflows are subjected to a more complex graph-based comparison.

In this respect, our previous results are encouraging: Next to retrieval quality of single similarity algorithms, we also investigated how multiple structure-based and annotation-based measures can be stacked and ensembled into combined similarity measures to benefit both result quality and retrieval speed. In particular, we have shown:

1. On the level of whole workflows, combining the use of a (structure-agnostic) Module Set approach for retrieval with a (structure-aware) Layer Decomposition step for reranking of the initial retrieval results maintains result quality in comparison to purely structure-based retrieval (Section 5.4.4)

2. For single module comparison, the edit distance of their labels can be effectively used to assess their functional similarity - for workflows where module labels are telling (4.4.1).

3. External knowledge derived from the repository not only improves result quality, but also reduces the sizes of the compared workflows, which leads to a speedup of the comparison process (4.4.1).

Inspired by these findings, we here present an approach for fast similarity search in scientific workflow repositories that takes the workflows' structure into account.

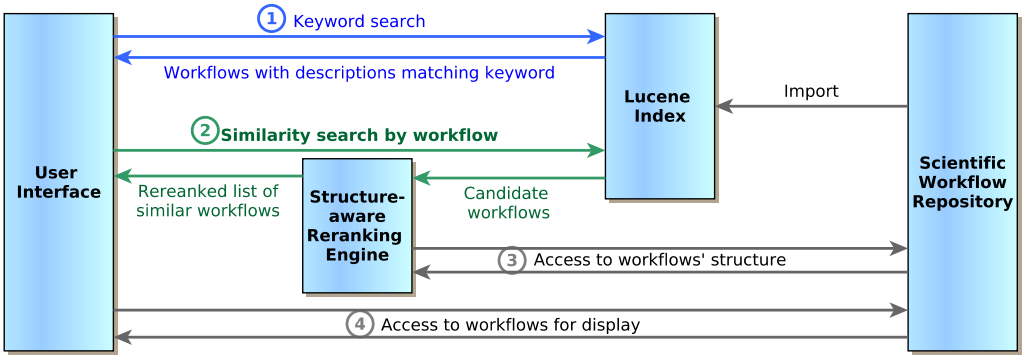


Figure 6.1: Schematic overview of scientific workflow similarity search using structure-based reranking.

The main goal is to demonstrate the feasibility of such a system, and to investigate transferability of our previous retrieval results to a repository-scale real-world scenario. In this chapter, we show how our previous findings can be leveraged to, a) efficiently index scientific workflows for fast similarity search using off-the-shelf technology, b) improve retrieval precision within the top-x results by reranking the initial structure-agnostic search results with the structure-aware Layer Decomposition algorithm, and c) further speed up the overall retrieval process by tweaking specific subtasks of the reranking algorithm.

In the following, we first give an account of our proposed architecture and how it indexes workflows. In Section 6.3 we evaluate the system for its retrieval quality and runtime. We conclude in Section 6.4.

6.1 Two-phase Retrieval Architecture

Taking from our findings on reranked retrieval of scientific workflows described in the previous chapter, we constructed a fast method for scientific workflow similarity search. We use a two-phased approach consisting of an initial, structure-agnostic retrieval step using off-the-shelf indexing technology, and a subsequent step of result reranking using more complex structure-based workflow comparison. An overview of the system as a whole is given in Figure 6.1. In the following, we describe the index, and the steps of retrieval and reranking in detail.

6.1.1 Workflow Indexing and Retrieval

The first step is to index workflow properties and modules for fast initial retrieval. As we have shown that module-based retrieval provides a very good preselection of search results, it is a natural target for indexing. Thus, we hypothesize that it is not necessary to apply more complex graph indexing databases and datastructures

such as neo4j [1], or the subgraph indexing system used in [43], but sufficient to use fast and simple approaches that capture the necessary details of the workflows we want to index: their modules. A search over such an index would specify a number of modules the query workflow contains, which would then be matched onto the sets of modules stored with each workflow in the index. The fundamental operation required from such an index is the ability to compare single modules. In Section 4.4.1, we found that module comparison can be effectively done based on the edit-distance of the modules' labels. This consideration of representing modules by their labels only, yet with the constraint of being able to search within these labels (i.e., strings) using approximate matching, suggests the use of well established document indexing systems for our purposes. A very renowned such system is Lucene [46].

Indexing Workflows in Lucene

Lucene is a Java-based document indexing and search engine that includes options for analyzing the documents prior to indexing (e.g., stop word removal), and for sophisticated ranking of search results by their (document-centric) relevance to the respective keyword query. In Lucene, *documents* fed to the index are composed of *fields*, where each field contains a sequence of *terms*. For full text documents, the words in the text naturally represent these terms; the text as a whole makes up one field; and the document as a whole may, next to the field representing its full text body, contain additional fields to hold, for instance, its title or the date of publication. From the terms stored, Lucene creates an inverted index linking each term to the documents and fields it is contained in. For searching, a given keyword query (i.e., one or more search terms) is matched in the index to find the corresponding documents. Lucene then ranks the matching documents by their relevance to the query. Next to strict matching of terms in the index, *fuzzy* searching is also supported. When performing a fuzzy search, Lucene uses the edit distance between terms in the query and terms in the index to find matching documents, which perfectly maps to our requirements for comparing module labels.

Following Lucene's document structure, we represent a workflow as a set of fields. One field holds the set of modules the workflow contains, with the modules' labels being the respective terms. This approach is the most straightforward setting for both indexing and search and, as we will see, provides surprisingly satisfying results (see Section 6.3.1). We discuss other options for representing workflows in Lucene in Section 6.4.

Apart from its apparent suitability for indexing of workflows by modules, using Lucene for workflow indexing provides an inherent benefit to our overall purpose of serving similarity search over scientific workflow repositories: Such a search often starts with an initial keyword query that roughly specifies the user's needs in terms of workflow functionality (see step (1) in Figure 6.1). Users then select one candidate workflow and let the system retrieve functionally similar workflows, i.e., the system

Field	Value
description	Given a specific entrez gene id, returns the pathways that this gene participates in and for each of those pa
id	2805
local filename	_http___www_myexperiment_org_workflows_2805_versions_4_
module count	13
modules	Clean_List_of_Strings_by_separator Get_image_from_URL SplitOnNewLine add_ncbi_to_string color_pathway_b
tags	kegg pathways entrez gene id pathway
title	Get Pathway-Genes by Entrez gene id
url	<http://www.myexperiment.org/workflows/2805/versions/4>

Figure 6.2: Document representation of myExperiment workflow 2805 in Lucene. See also Figure 4.1 on page 50.

performs a workflow similarity search. As such, next to our primary interest of making the workflows searchable by the modules they contain, for practical reasons such an index would also contain a workflow's title, description, and tags associated to it - in which a user's keyword queries would be matched.

Figure 6.2 shows how a workflow is stored in the index using these fields. Next to the aforementioned fields for the workflow's modules, its title, description, and tags, we also store the workflow's id and url in the original repository (here myExperiment), the filename of the workflow definition file in our system, and the number of modules the workflow contains. Note that these additional fields are merely informative, and are currently not used for searching. The tabular view of fields and values in Figure 6.2 is taken from Luke[1], an index analysis tool for Lucene. Using this tool, Figure 6.3 gives a global view on the index, summarizing some of its properties: The right, lower hand pane shows the most frequent single terms contained in the index together with the fields these are stored in. This reveals that 186 of 1483 workflows in our dataset have a single module only, and (not shown in the figure at position 28) that the most frequently used module label is *split_string_into_string_list_by_regular_expression* with 140 occurrences - which well aligns with our findings from the repository analysis performed in Chapter 3 (see Table 3.2).
The left, lower hand pane shows the fields our workflow documents contain, how many distinct terms are indexed for each of them, and what their relative share of the index is. Note that in terms of size, the index uses only 1.7MB on disk. Creating the index for our repository of 1483 Taverna workflows takes approximately 4 minutes.

Searching the Index

Lucene provides its own query syntax for searching the index. Most important to us is the ability to specify over which fields a search is to be carried out, and what the maximum edit distance is at which Lucene will consider terms to match. In Lucene's terminology this distance is given by the minimum similarity at which its

[1] http://code.google.com/p/luke/

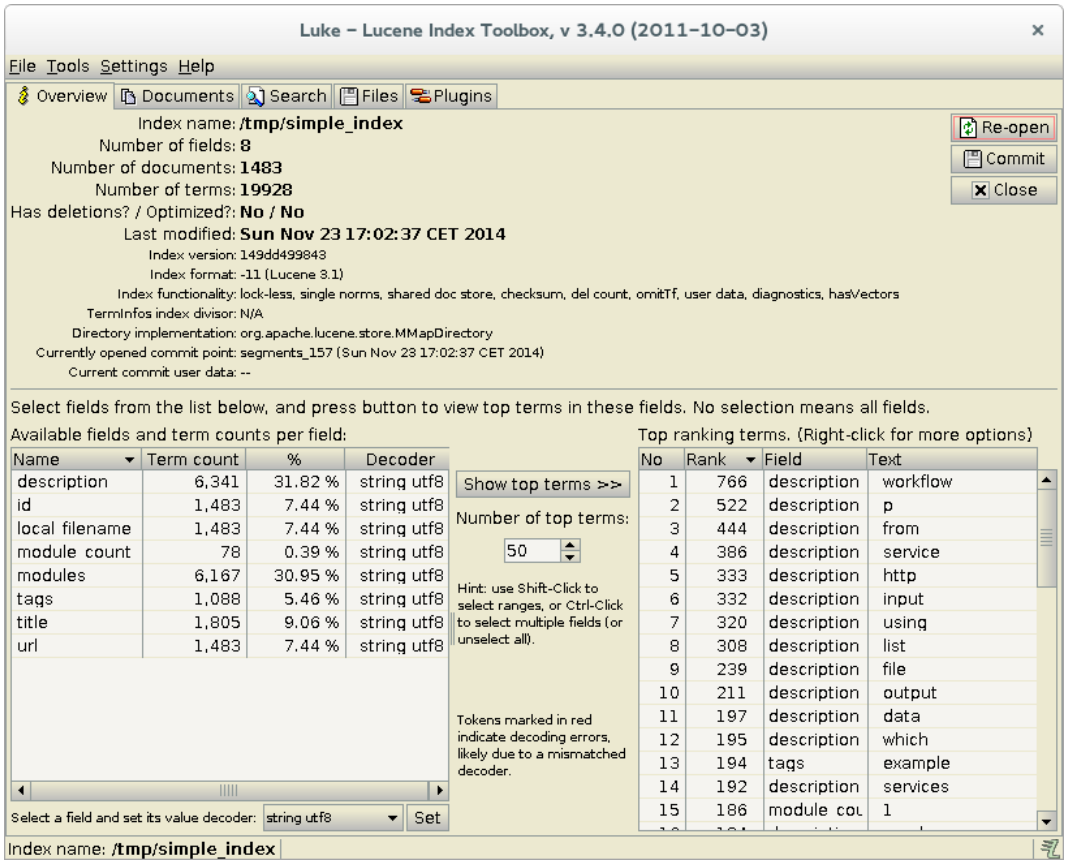


Figure 6.3: Overview of Lucene workflow index in Luke index diagnosis tool.

Listing 6.1: Example fuzzy query to the Lucene index constructed from the modules of myExperiment workflow 1189 (see also Fig. 4.1, page 50).

```
(modules:Filter_list_of_strings_by_regex~0.7)
(modules:Get_image_from_url~0.7)
(modules:Split_string_into_string_list_by_regular_expression~0.7)
(modules:bconv_2~0.7)
(modules:get_pathways_by_genes~0.7)
(modules:mark_pathway_by_objects~0.7)
(modules:regex_value~0.7)
(modules:regex_value_1~0.7)
```

fuzzy query processing may consider a pair of terms to match. It takes values between 0 and 1, and corresponds to the number of allowed edit operations wrt terms' length. In the following, we refer to this minimum similarity as *fuzzyness*.

To perform a workflow similarity search by modules, we construct a query from the labels of the modules of a given query workflow. An example query is shown in Listing 6.1. For each module the *modules* field is specified to be searched in, and the desired fuzzyness is appended to the module's label (separated by a ~). This example query returns a total of 289 results.

6.1.2 Structure-based Result Reranking

After fast structure-agnostic retrieval of candidate results, we rerank these results by the structure-aware Layer Decomposition algorithm. In our previous evaluation (see Section 5.4), this algorithm has not only shown to provide best results in the task of workflow ranking, but to also be comparatively fast.

Layer Decomposition Configurations

In the qualitative evaluation provided in the previous chapter, our focus was on using the Layer Decomposition algorithm with those settings that require as little background knowledge about the repository it was to be deployed in as possible - even when the inclusion of such knowledge provided better results. When creating a search engine for one specific repository of scientific workflows, on the other hand, extensive background information and tuning will be used to deliver a better user experience. To contrast these perspectives of off-the-shelf applicability and customizability, we here use the Layer Decomposition approach in two settings:

- *LD_np_ta_pll* does not use any knowledge of the repository and its workflows, apart from the assumption that the modules' labels are telling of their functionality - a reasonable assumption in our setting given the way our index is constructed.
- *LDpml25_ip_te_pll* uses the maximum amount of knowledge we have, including *importance projection* to filter out unspecific modules, *type equivalence*

Table 6.1: Configurations of the Layer Decomposition algorithm used for reranking, and the impact the corresponding tweaks have on algorithm quality (Q) and speed (S). (+: improved, o: unchanged)

Comparison step	LD_np_ta_pll	LDpml25_ip_te_pll	Benefit	
			Q	S
Workflow Preprocessing	np: no projection	ip: importance projection filtering out unspecific modules.	+	+
Module Comparison	ta: no preselection of pairs for comparison, all module pairs are compared. pll: only the *labels* are compared by edit distance	te: preselection of modules pairs for detailed comparison by *type equivalence*. pll: only the *labels* are compared by edit distance	o	+
Module Mapping	maximum weight matching	maximum weight matching		
Topological Comparison	LD: Layer Decomposition	LDpml25: Layer Decomposition, penalizing layer mismatch exceeding 25% of the layers in the larger workflow	+	o
Normalization	Jaccard variation	Jaccard variation		

module pair preselection to reduce the number of detailed module comparisons made (see Section 4.1.1), and *penalties for mismatching layers* in the Layer Decomposition topological comparison itself (see Section 5.4.2).

Table 6.1 lists each step of the workflow comparison process and how it is treated in these two configurations (and decodes the intricate notation). It also shows how tuning at each step affects quality and speed of the algorithms, as shown in the corresponding evaluations in Sections 4.4.1 and 5.4.2. We will see how this translates to concrete runtimes of workflow comparison subtasks in Section 6.3.2.

Ensembles of Structure and Annotation

Next to the performance of single algorithms, we have also investigated how multiple algorithms can be combined into *ensembles* (see Section 4.4.1). These ensembles allow to integrate different perspectives on workflow similarity, especially those provided by structure-based and annotation-based comparison. How applicable such ensembles are to any given repository depends on the amount of annotations it provides. In Section 2.2.2, we illustrated that current repositories greatly differ in this respect, and in Section 4.4.3 we found that the lack of textual descriptions of workflows greatly affects the applicability of annotation-based measures to the Galaxy public workflow repository. When annotations are available, on the other hand, we have shown the use of ensembles to greatly benefit result quality in similarity search. We thus include an option of using the ensemble of *Bag of Words* similarity (over the workflows' titles and descriptions, see Section 4.1.2) and Layer Decomposition in either of the configurations listed above. Ensemble similarity values are derived as mean average of the values computed by its constituting algorithms.

6.2 Related Work

While previous work has investigated several options to determine similarity of scientific workflows, e.g, [78, 76, 82, 8, 43, 39, 88] (see Chapter 4), only litte work exists that targets their application to fast similarity search over whole repositories. A system using subgraph matching to search a repository of scientific workflows using a given query workflow is proposed in [43]. The authors report runtimes of 15 seconds over a repository of 89 workflows and manually evaluate retrieval quality on an example workflow (explicitly rejecting to derive generic claims). In [9], a system for similarity search over workflow repositories is proposed that uses manually added semantic annotations on cooking workflows and their components for comparision. While such annotations are not found on workflows in current scientific workflow repositories, the architecture of the proposed system is analogous to the two-phased approach introduced in the previous section: An initial set of candidate results is retrieved by a fast search using an index over the semantic annotations the workflows contain. These candidates are then reranked by a graph-based approach that determines workflow similarity as the maximum aggretage similarity of the nodes (i.e., modules) contained in a respective mapping between two workflows – similar to the Module Sets approach introduced in 4.1. While runtime is shown to clearly benefit from the two-step approach, retrieval quality of the compound system is evaluated against the results retrieved by the standalone graph-based approach only, limiting general interpretability of results. For business workflows, [89] target a similar, two-step approach, specifically addressing the first step of candidate retrieval: A selection of features derived from the graph structure of the workflows is used to index the workflows in a repository. For a given query workflow this index is used to provide and estimate of whether a workflow from the repository is 'irrelevant' to a search or 'potentially relevant'. The candidates from the latter class are then be process by more complex similarity measures for business workflows, e.g., [34, 32, 33]. Analogously, the results returned by the initial search over the Lucene index used by our system could be reranked using any available similarity measure for scientific workflows (see above). We use Layer Decomposition for its superiority in previous evaluations.

6.3 Evaluation

We evaluate our system for scientific workflow similarity search on the corpus of scientific workflows introduced in Chapter 4, Section 4.3. While the myExperiment repository the corpus originates from has grown in size since the corpus was created, for comparability of results to our previous findings we explicitly limit our evaluation to the workflows contained in this corpus. The corpus consists of 1483 Taverna workflows from the myExperiment repository. For 485 pairs of scientific workflows from this corpus, a total of 2424 similarity ratings were provided by 15 human experts. This includes a set of 8 (query) workflows, for which similarity ratings are

available which cover all workflows returned by a similarity search over the whole corpus by a selection of similarity algorithms. We use and extend these ratings for evaluation of retrieval quality by the system proposed here.

In the following, we first comparatively investigate how different settings and ensembles of algorithms affect the quality of the retrieved results, both confirming our previous findings, and even improving on them. In Section 6.3.1 we then measure how runtime of similarity search over the whole dataset (i.e., a whole repository) is affected by indexing and different settings discussed in the previous section.

6.3.1 Retrieval Quality

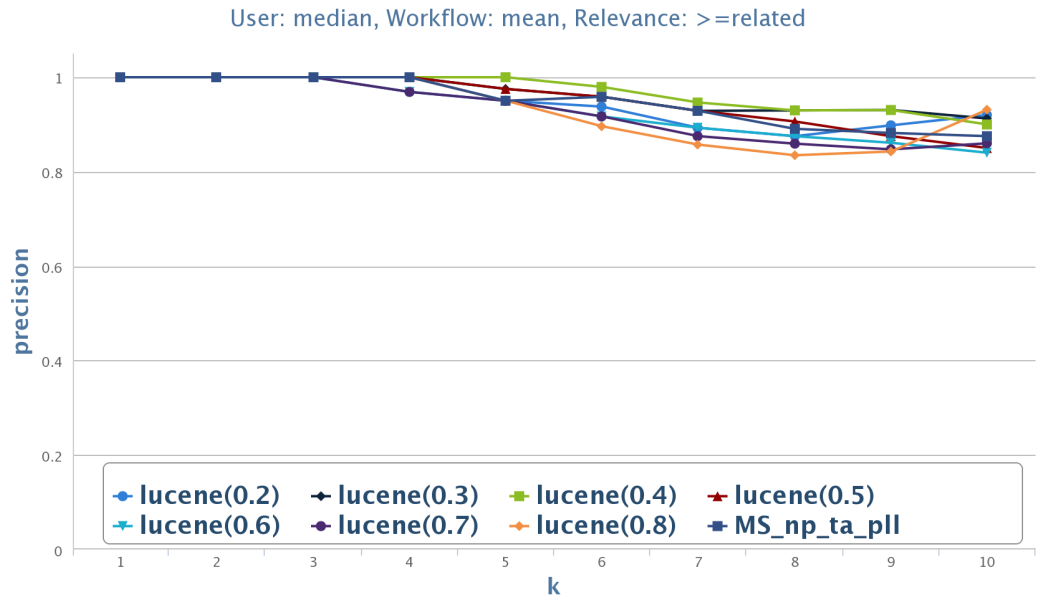
Analogous to the evaluations performed in previous chapters (Sections 4.4.2 and 5.4.2), we measure retrieval quality as the precision at k over the top 10 results retrieved by a similarity search over the whole repository. Relevance of a search result is determined by the median expert similarity rating assigned to each pair of query and result workflow in the corpus. Recall that expert ratings where given along a five step Likert scale (*very similar, similar, related, dissimilar, unsure*). We thus evaluate precision at k with the relevance thresholds of *similar* and *related*. *very similar* results are found by all presented algorithms at near equal quality (data not shown).

In our system, retrieval quality is influenced by three factors: (i) the minimum similarity of labels (fuzzyness) set for module label matching in the index, (ii) the number of results retrieved and fed to the reranking step, and (iii) the algorithm(s) and their configurations used for reranking.

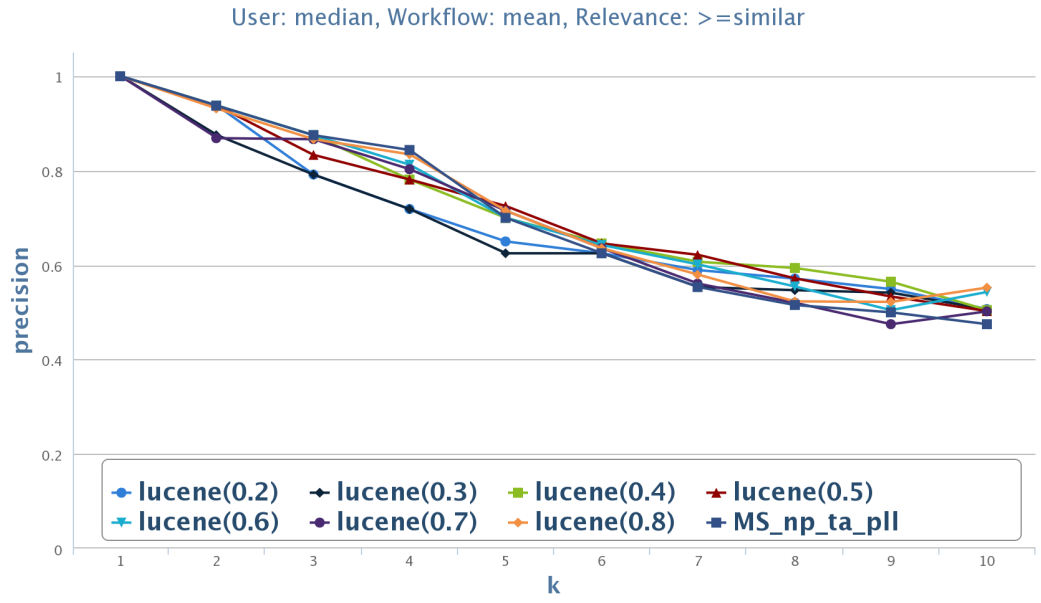
Minimum Similarity of Module Labels

Figure 6.4 shows precision at k for the top 10 results retrieved by Lucene with various settings of fuzzyness, in direct comparison to the Module Set algorithm used in a comparable configuration: all of the workflows' modules are used for similarity assessment (no *ip* or *te*) and module labels are compared by edit distance (*pll*). Retrieval quality of Lucene is clearly on par with Module Sets. Most obviously, different fuzzyness values greatly influence the results returned. Observe that for a relevance threshold of *similar* (Fig. 6.4b) there is an apparent trend of higher values to deliver better results: While values of 0.2 and 0.3 are clear outliers for the negative, values of 0.4 and above provide comparable results over the first 5 positions. For the second half of the top 10, values above 0.7 appear too strict, and are outperformed by more fuzzy matching of lower minimum similarity values. This observation of strict versus fuzzy matching is confirmed at a relevance threshold of *related* (Fig. 6.4a), where especially values of 0.3 and 0.4 provide convincing results.

As we are ultimately interested in reranking the results retrieved by Lucene with any such setting, another aspect to consider is the number of results Lucene retrieves for each value. We searched our index with each of its 1483 workflows in turn, using different settings for fuzzyness of label matching. Table 6.2 reveals that values



(a)



(b)

Figure 6.4: Mean retrieval precision at k against the median expert rating for Lucene with various settings for the minimum similarity of module labels, and Module Sets workflow comparison for relevance threshold (a) related, and (b) similar. Module Sets used with module similarity by edit distance of labels (*pll*), without *ip* and *te*.

Table 6.2: Numbers of query workflows out of 1483 for which less than 10, 25, and 50 results could be retrieved using the respective *minimum similarity* values.

Lucene minimum similarity	No of query workflows yielding		
	< 10 results	< 25 results	< 50 results
0.2	1	12	27
0.3	21	59	114
0.4	90	185	313
0.5	208	356	556
0.6	298	506	687
0.7	382	624	809
0.8	472	683	891
0.9	498	713	908

above 0.4 result in difficulties in retrieving sufficient numbers of workflows. As a consequence, we focus further evaluation on minimum similarity values of 0.2, 0.3 and 0.4.

Limiting reranking candidates

While for some workflows only limited lists of candidates are available, the majority of query workflows yields large lists of candidate results. For instance, with a minimum similarity value of 0.3, about half of the workflows in the repository have more than 300 candidates (one third for 0.4). We are thus interested in reducing the number of candidates for the reranking phase to only the top-x, i.e., a specific ranking cutoff. For a fuzzyness of 0.3, Figure 6.5 shows retrieval precision at k for reranking at different such cutoffs. We use the reranking algorithm which has, so far, proven to provide best results (to be comparatively evaluated in the next section). While all cutoffs provide rather similar performance, reranking of the 25 topmost candidates seems to deliver slightly higher result quality. First and foremost, these findings show that quality of reranked retrieval does not necessarily increase with the number of candidate results used for reranking. Apparently there is a synergetic effect of the candidate preselection done by Lucene (i.e., workflow comparison by modules only), and the more complex reranking.

For fuzzyness values of 0.2 and 0.4 results are equivalent, only that their peaks in performance are not at cutoffs of 25, but 35 and 15, respectively (data not shown). This shows that the differences in retrieval quality observed for different fuzzyness values in Lucene-only retrieval in the previous section, are evened out by the applied reranking. Together with the observations on the numbers of results returned by Lucene in the first place (see Table 6.2), it also indicates a trade-off between result quality, the amount of reranking required to achieve it, and the overall number of results available to the user: While with less fuzzy retrieval only the top 15 results have to be reranked to provide best results amongst the top-10, a significant portion of query workflows will not see 10 results at all. The more fuzzy initial retrieval is, on the other hand, the more query workflows will yield larger result sets, but the more reranking has to be applied. Which settings to use depends on the resources

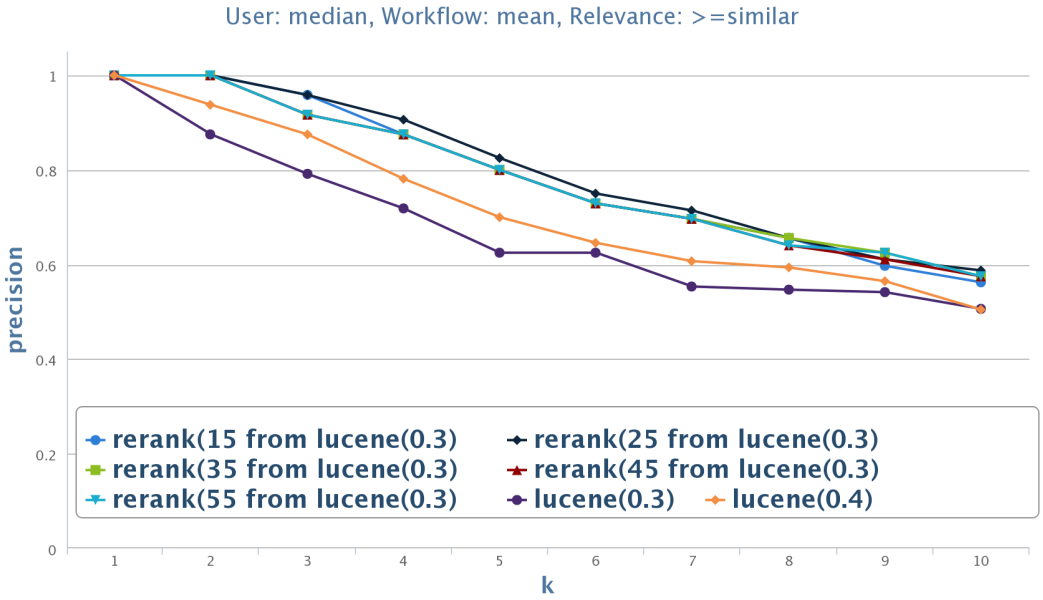


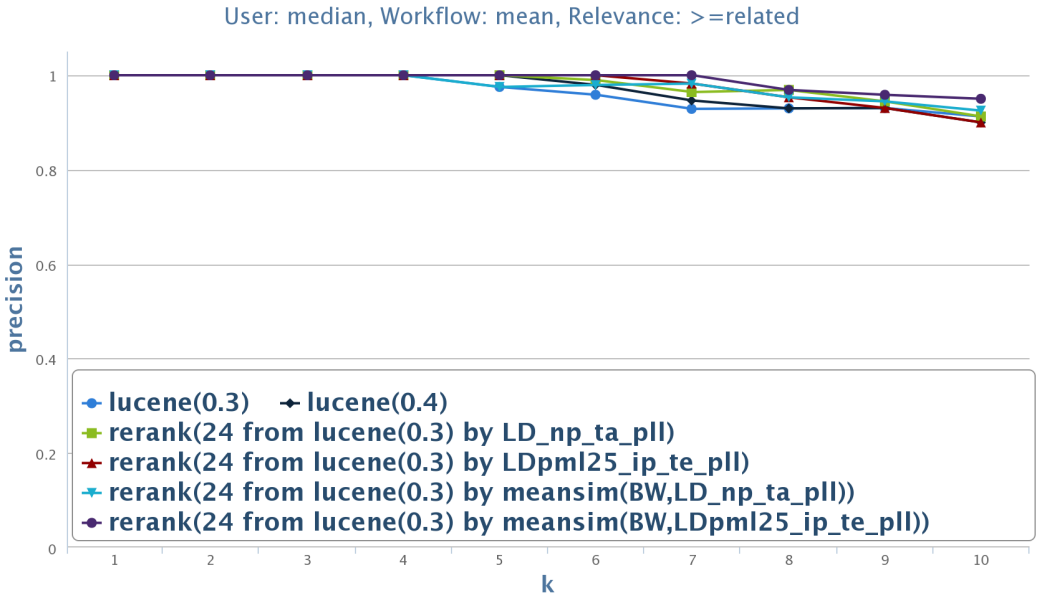
Figure 6.5: Mean retrieval precision at k against the median expert rating for reranking of different numbers of top-x candidates retried by Lucene with fuzzyness 0.3. Lucene standalone retrieval included as baseline. Relevance threshold similar.

available for reranking.

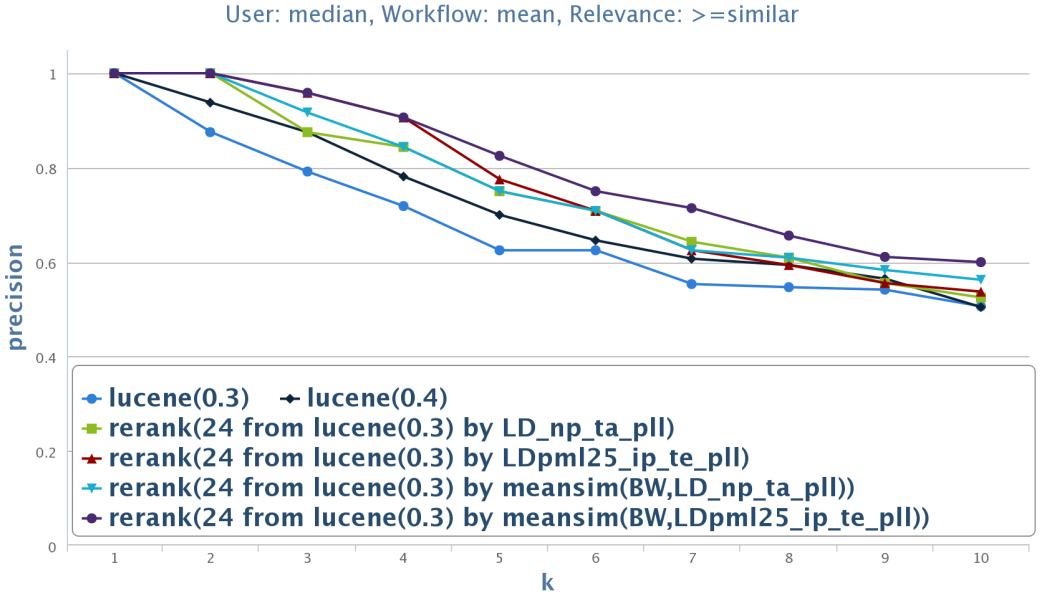
Based on these considerations, all further evaluations use reranking of the top 24 results retrieved using Lucene with a fuzzyness of 0.3, best matching the resources used for runtime evaluation in Section 6.3.2.

Reranking algorithms

Figure 6.6 shows how result quality is affected by the two contrary configurations of Layer Decomposition introduced in Section 6.1.2. The figure also includes the ensembles of these two configurations with the annotation-based measure of *Bag of Words (BW)*. Results correlate at both relevance thresholds of related and similar (Fig. 6.6a and b, respectively). Clearly, both the amount of repository knowledge included and the use of ensembles benefit overall retrieval precision: The best reranking approach is the ensemble of fully tuned Layer Decomposition and Bag of Words. The ensemble including the naive configuration of Layer Decomposition is still outperformed by the standalone, tuned Layer Decomposition when it comes to the very top of the result list, and performs equally well as the standalone naive version. Most importantly, all reranking methods deliver better results than non-reranked retrieval by Lucene only.



(a)



(b)

Figure 6.6: Mean retrieval precision at k against the median expert rating for Lucene's top 24 results reranked by different (ensembles of) algorithms for relevance threshold (a) related, and (b) similar.

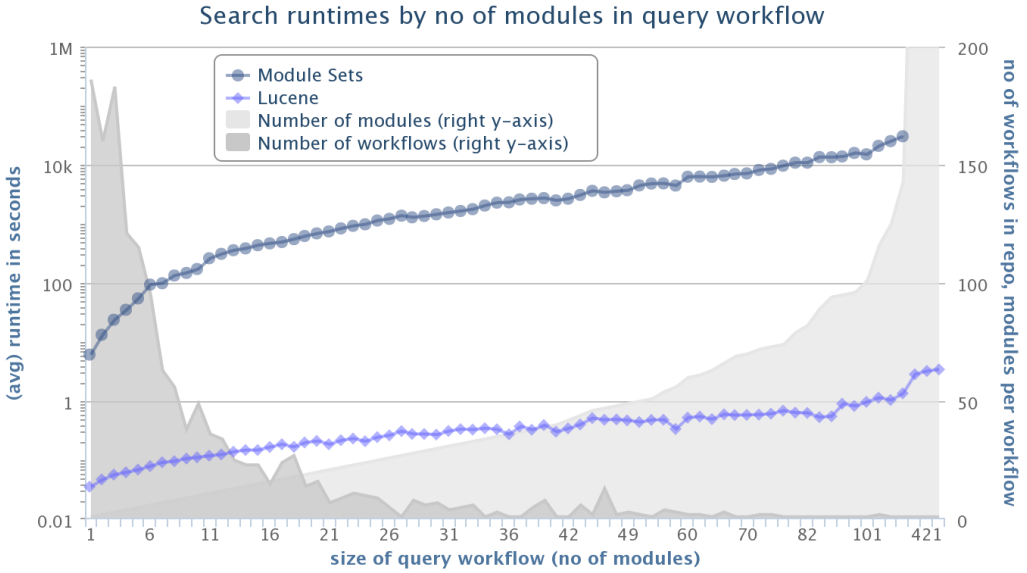


Figure 6.7: Average runtime (left logarithmic y-axis) of similarity search over repository of 1483 workflows using Module Set workflow comparison (MS) or Lucene, by number of modules in query workflow. Areas (right, linear y-axis): No of workflows in repository with the corresponding number of modules; No of modules (as per value on x-axis)

6.3.2 Runtime

We measure the time taken for retrieval of the top 10 most similar workflows from the whole repository using a given query workflow. For Lucene, we use the default configuration of single threaded search over its index. For structure-based workflow comparison, e.g., retrieval by Module Set comparison or reranking by Layer Decomposition, a setup of 24 parallel processes is used, each handling the comparison of one pair of query workflow and workflow from the repository at a time. While this is a reasonable setup in a server-based environment, it has to be kept in mind that fully sequential comparison of all pairs of workflows would take substantially longer. Yet, again, when less resources are available, a different setting of initial retrieval fuzzyness and reranking cutoff might be used.

Search Phase

Figure 6.7 plots runtimes of similarity search over the whole repository for the Module Sets similarity algorithm and Lucene against the number of modules in the query workflow (again, note that Module Set comparisons are parallelized 24-fold). For small workflows, Module Sets is in fact respectably fast, given the fact that it does not make use of any index structures but has to compare each pair of query

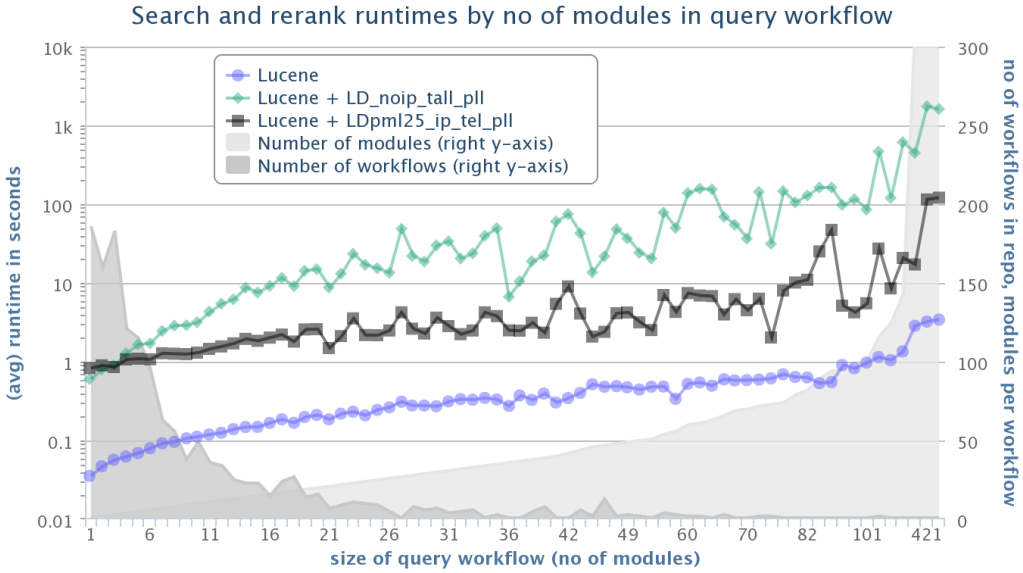
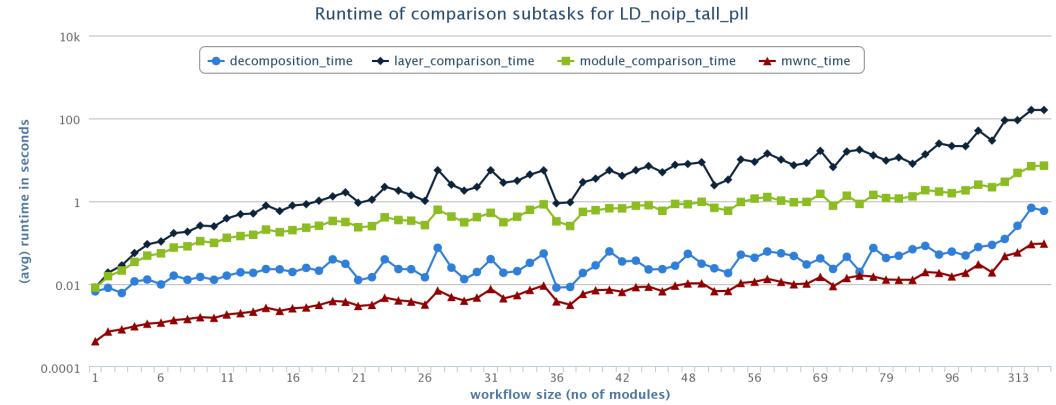


Figure 6.8: Average runtime (left logarithmic y-axis) of similarity search over repository of 1483 workflows using Lucene and reranking of top 24 candidates by either of two configurations of Layer Decomposition (see text); plotted by number of modules in query workflow. Areas (right, linear y-axis): No of workflows in repository with the corresponding number of modules; No of modules (as per value on x-axis)

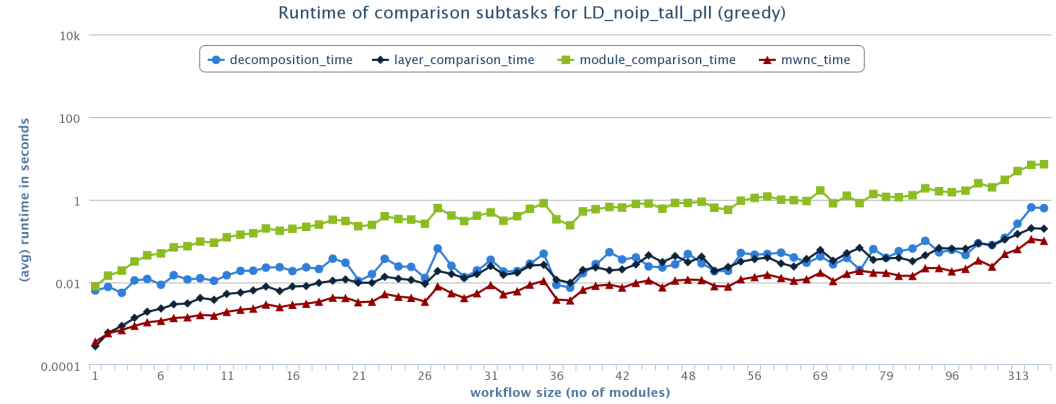
workflow and repository workflow separately. With only 5 modules in the query workflow, search times average at one minute already, and double once more at 8 modules. Not surprisingly, Lucene, one the other hand, is faster by several orders of magnitude, most probably not only due to its indexing, but also due to its implementation. Only for very large workflows do retrieval times reach or exceed one second. Observe that workflow sizes on the x-axis don't increase linearly - for a better visual grasp, we also plot the x-values themselves, showing a steep increase of workflow sizes towards the right end of the plot. Furthermore, for a better feel of how the retrieval times translate to the repository's contents, the darker area graphs the number of workflows contained in the repository which have the corresponding number of modules. Clearly, most workflows in this specific repository are rather small, which corresponds to the average workflow size of 11.4 modules reported on in Chapter 3, Section 3.1.1.

Reranking Phase

After retrieval of candidates from the index we rerank the top 24 results. Figure 6.8 shows runtimes of both configurations of Layer Decomposition. The tuned configuration is faster by an order of magnitude. Note that for clarity's sake we don't



(a)



(b)

Figure 6.9: Runtimes of subtasks of Layer Decomposition workflow comparison in dependence of workflow size; (a) with maximum weight matching of modules for assessing similarity of single pairs of layers, and (b) with greedy mapping.

show runtimes of ensembles - including Bag of Words in reranking has practically no effect on runtimes.

The advantage of the tuned algorithm over the naive one is a consequence of the speedups listed in Table 6.1. While these speedups already lead to overall similarity search times of less than 10 seconds for the vast majority of workflows in the repository (reranking only the top 24 results), our interest is to see exactly which portion of the Layer Decomposition algorithm is taking how long, and if further speedup is possible that does not include repository knowledge. Dissecting the workflow comparison of Layer Decomposition, Figure 6.9a reveals that the major part of time is spent comparing all pairs of modules (per edit distance), and comparing all pairs of layers. The times taken for decomposing a workflow into its layers in the first place, and for computing the maximum weight non-crossing matching of those layers after their pairwise comparison are much lower.

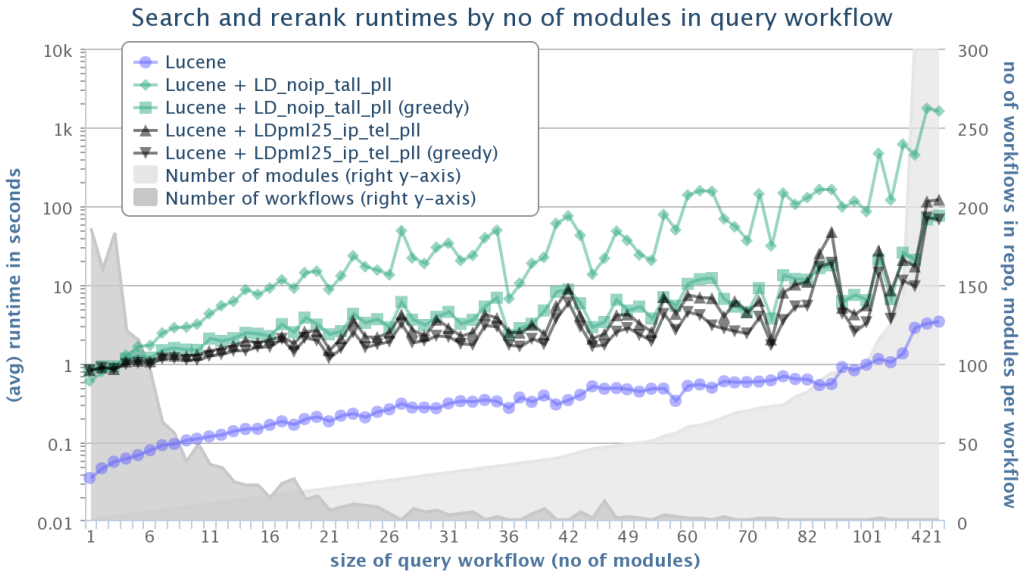


Figure 6.10: Average runtime (left logarithmic y-axis) of similarity search over repository of 1483 workflows using Lucene and reranking of top 24 candidates by either of two configurations of Layer Decomposition (see text); plotted by number of modules in query workflow. Areas (right, linear y-axis): No of workflows in repository with the corresponding number of modules; No of modules (as per value on x-axis)

Recall from Section 5.3 that Layer Decomposition uses the *maximum weight matching* of the sets of modules two layers are composed of to determine layer-wise similarity. The algorithmic complexity of computing such a matching of two layers L_1 and L_2 is $O((L_1 \cup L_2))^2 L_1 L_2)$. Most interestingly in this respect, we have shown in Section 4.4.1 that for workflow comparison by Module Sets, using greedy mapping of modules instead of maximum weight matching provides equivalent results. Applying this finding to the layer comparison step of Layer Decomposition yields a significant speedup both of the time taken for this specific step, apparent from Figure 6.9b, and of the time taken for similarity search as a whole: Figure 6.10 reiterates our previous illustration with additional runtimes plotted for the greedy versions of the reranking algorithms. Especially for the zero-knowledge configuration, the improvement is substantial. And, while less accentuated, the fully tuned version of Layer Decomposition benefits from the shift in complexity as well, now yielding overall search times under 5 seconds for most workflow sizes. Most importantly, retrieval quality remains unchanged.

6.4 Conclusion

In this chapter we introduced a system for structure-aware similarity search in scientific workflow repositories. Relying on off-the-shelf indexing technology in form of Lucene and using a two-phase approach of candidate retrieval and reranking, we were able to apply the high quality structure-based Layer Decomposition workflow similarity measure at repository-scale with acceptable speed. We distinguished two cases of application, where knowledge about the repository and its workflows is available for fine-tuning of structure-based comparison - or not. For both of these cases we showed that reranking of results clearly improves retrieval quality of similarity search, and closely investigated their runtime properties. The speedup achieved by greedy assessment of layer similarity in workflow comparison even for the naive version of Layer Decomposition is promising for the application of this method to repositories where external knowledge such as for importance projection or layer mismatch penalty are not available.

Still there is room for further improvement: our evaluation showed that the most costly step for Layer Decomposition runtime is now single module comparison, which is currently based on the Levenshtein edit distance of the modules' labels. While Lucene is very fast at computing this distance over its index for the query terms, our implementation clearly lacks speed. Next to improving on this implementation itself, a more twisted option might be to delegate layer-wise comparison to Lucene just as well, indexing not only whole workflows but their layers. Taking this consideration one step further, recent versions of Lucene (starting at version 3.4) also provide the ability to model interrelations between documents. As such, both the workflow with its global properties, and each single one of its modules could be indexed separately, and then be linked in a search to potentially express structural constraints in the initial query already. We leave further thought on this issue to future work.

On the more conceptual level, regarding the three parameters of fuzzyness of initial candidate retrieval, top-x cutoff of candidates fed to the reranking step, and reranking method to set for optimizing retrieval quality, we found that for the first two, no single best setting exists for best result quality, but the most appropriate setting depends on available computational resources. In this respect, another interesting option may be to not select a fixed number of candidates to rerank, but to use a dynamic cutoff based on the similarity values found during reranking. For the last parameter, the reranking method to apply, a clear trend can be observed: The more knowledge about a repository and its workflows is available, and the more different perspectives on workflow similarity can be applied, the better. Especially the latter point of different perspectives puts a hard eye on future research to access other sources of workflow meta-data for similarity assessment, one such source being provenance.

7 Summary and Outlook

In this chapter we conclude the thesis. The summary of our main contributions and findings in Section 7.1 is followed by an outlook on future directions of research in Section 7.2.

7.1 Summary

In this work we presented a comprehensive investigation of similarity measures for scientific workflows, motivated by the wish to advance discoverability of workflows in scientific workflow repositories. While scientific workflow management systems have gone a long way in supporting workflow authors in the creation of these workflows, providing proper tools and mechanisms for the discovery of existing workflows in public repositories, may make these workflows accessible to an even broader audience. The current state-of-the art in supporting the discovery of scientific workflows relies on annotations to be provided for these workflows by their authors, and for the potential re-users of these workflows to use the right keywords when searching for them - both of which is often not the case. To improve this situation, our research followed four consecutive steps, starting from a detailed analysis of workflows in public scientific workflow repositories, and ending at the implementation of a system for fast similarity search in these repositories, making use of the workflows' structure. We made a number of contributions, which we summarize in the following.

The first contribution were the findings derived from said analysis of workflows and their elements in public repositories. We carefully identified single modules across different workflows (and authors), and, following an existing classification of module types into (rather technical) functional categories, we were able to characterize which modules are used most frequently across workflows, and which are more distinctive of workflow functionality.

We made direct use of this knowledge in our second step, where we introduced a framework for deeply comparing existing approaches to structure-based comparison of scientific workflows. Next to dissecting and re-implementing a number of existing approaches following the workflow comparison process set up in the framework, we also included options for preprocessing the workflows before comparison, based on the knowledge taken from the repository. For the evaluation of the methods so implemented, one of the most important contributions of this thesis is the collection of a sizable gold-standard corpus of human-provided workflow similarity ratings. For the first time, corpus and framework allowed a systematic, comparative evaluation of various scientific workflow similarity measures, both annotation-based and structure-based. In this evaluation, we pinpointed each step of workflow comparison

and found that a) for module comparison, the label given to the module by its author conveys enough information about the module's functionality to assess module similarity from the edit distance of these labels - whereas strict matching of these labels used in previous work was not sufficient; b) for topological comparison, there is a gradient of tunability and quality being highest for comparison based only on sets of modules, and lowest for comparison of full structure by graph-edit-distance - with a stable (but computationally slow) balance at the level of substructures; and d) while structure-based comparison tuned with repository derived knowledge clearly outperforms annotation-based measures, ensembles of both of these types of comparison provide best quality results.

Taking directly from these findings, and targeting a sweet-spot between substructure-based and module-only comparison of whole workflows, we proposed Layer Decomposition as a novel structure-aware scientific workflow similarity measure. Evaluation within our framework showed that this approach provided best results at higher speeds than traditional structure-aware methods - yet still being too slow for user-friendly application to similarity search over whole repositories. Addressing this issue, we first extended our findings on combining different algorithms into ensembles by additionally stacking them for initial retrieval and following reranking of workflows in similarity search. We then translated our findings into a fast, working search engine for structure-based similarity search at repository-scale.

In this final contribution, we used off-the-shelf document indexing technology to provide fast candidate retrieval from module-based similarity search. These candidates are then reranked by the Layer Decomposition algorithm, or, better yet, by its ensemble with annotation based comparison. Evaluation showed that this system provides very high quality results at high speed - not only, but also due to the inclusion of the knowledge derived from our starting contribution. We thus hope to integrate our system with a public online scientific workflow repository to continuously improve scientific workflow discovery.

7.2 Future Directions

Our results point to various future directions. Starting with the most immanent potential for future work, several options exist for further improving or extending our similarity search system. These options include the exploration of more complex indexing schemes to delegate subtasks of the structural comparison process to proven libraries (e.g., indexing of workflows' layers extracted by the Layer Decomposition algorithm, next to whole workflows), or the evaluation of strategies to dynamically determine the cutoff of candidates used for reranking.

More interestingly, other types of scientific workflow discovery will need to be targeted. For instance, similarity search using partial workflows could greatly benefit workflow authors by providing suggestions for extensions or completions of the (partial) workflows they are designing. Or, using similarity between workflow substructures, frequent patterns could be mined from the repository to provide both

functional subunits to be used in workflow design, and best-practice patterns to guide workflow authors [40]. In this thesis, our focus was drawn to similarity search for whole workflows - as one possible application of similarity measures for scientific workflows [21]. While we started to explore such measures in a rather unbiased fashion, the construction of our corpus of similarity ratings determined a certain type of setting for evaluation. Creating such extensive corpora for other types of discovery tasks, such as clustering, matching of partial workflows, or the identification of functional sub-units, is an important step towards the evaluation of corresponding algorithms - including our own. Potentially, the corpus collected in this thesis could even be re-used for creating, or bootstrapping others.

As we have investigated similarity measures for scientific workflows using annotations and structure, other sources of information are becoming available, which could be useful to assist workflow comparison. The possibility of using provenance for this purpose has already been discussed in Section 2.3.2. To include measures based on such new sources in the discovery process, investigation of ensembles such as the ones touched in this thesis will be necessary to explore more elaborate techniques [62].

Along the same lines, another area of important future work it the investigation of automatically derived knowledge from a repository about the workflows it contains. We have seen that such knowledge can greatly benefit the performance of functional comparison of scientific workflows, but is specific to repository and breed of workflows, and time-consuming to collect manually. How such knowledge can be transferred to other workflow formats and repositories, or how it can be collected in an automated way for a given repository, will have to be explored to provide optimal results in scientific workflow discovery.

Ultimately, the vision is to provide best possible support for users in discovering the workflows they need at any given point of data analysis and data processing. Ideally, a scientist user would load their data into an in-silico research tool (say, a SWFM) and immediately be provided with options for analyzing, augmenting, and transforming this data based on its type - matched against the workflows in a large repository. Selecting one such analysis option (or multiple to run in parallel in a cloud environment), the scientist would iteratively explore and analyze the data, always stepping from one data item to the next, transformed by scientific workflows, executed at only the click of a button. Whenever more than a handful of workflows exist for a next step, clustering of workflows would provide an easily browsable, drill-down interface for selecting the desired analysis. The resulting complete, iteratively designed in-silico experiment (read: scientific workflow) could then be stored and published online for re-use by the scientist him- or herself or by others. When uploading the workflow to a repository, annotations such as keyword tags would automatically be suggested based on the tags assigned to similar (as a whole or partially) workflows in the repository, to further facilitate the workflow's discovery.

A system as the one envisioned here, has more to it than similarity measures for scientific workflows, of course, but to make it possible, each of the research directions outlined above will have to be explored.

Bibliography

[1] neo4j graph database. *http://neo4j.com/*.

[2] ProvBench initiative. *https://sites.google.com/site/provbench/*.

[3] SHIWA workflow repository. *http://shiwa-repo.cpc.wmin.ac.uk*.

[4] C. C. Aggarwal and H. Wang. Graph data management and mining: A survey of algorithms and applications. In *Managing and Mining Graph Data*, pages 13–68. Springer, 2010.

[5] D. Aumüller and A. Thor. Mashup-Werkzeuge zur Ad-hoc-Datenintegration im Web. *Datenbank-Spektrum*, 8:26, 2008.

[6] Z. Bao, S. Cohen-Boulakia, S. B. Davidson, A. Eyal, and S. Khanna. Differencing provenance in scientific workflows. *ICDE*, pages 808–819, 2009.

[7] A. Barker and J. Van Hemert. Scientific workflow: a survey and research directions. In *Parallel Processing and Applied Mathematics*, pages 746–753. Springer, 2008.

[8] R. Bergmann and Y. Gil. Similarity assessment and efficient retrieval of semantic workflows. *Information Systems*, 40:115–127, 2012.

[9] R. Bergmann, M. Minor, S. Islam, P. Schumacher, and A. Stromer. Scaling similarity-based retrieval of semantic workflows. In *ICCBR-Workshop on Process-oriented Case-Based Reasoning, Lyon*, pages 15–24. Citeseer, 2012.

[10] T. Berners-Lee and J. Hendler. Publishing on the semantic web. *Nature*, 410(6832):1023–1024, 2001.

[11] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su. Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. In *Astronomical Telescopes and Instrumentation*, pages 221–232. International Society for Optics and Photonics, 2004.

[12] P. Bertoli, M. Pistore, and P. Traverso. Automated composition of web services via planning in asynchronous domains. *Artificial Intelligence*, 174(3):316–361, 2010.

[13] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi. Characterization of scientific workflows. In *Third Workshop on Workflows in Support of Large-Scale Science (WORKS) 2008.*, pages 1–10. IEEE, 2008.

[14] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In *Conceptual Modeling–ER 2005*, pages 369–384. Springer, 2005.

[15] A. Brogi, S. Corfini, and R. Popescu. Semantics-based composition-oriented discovery of web services. *Transactions on Internet Technology*, 8(4):19, 2008.

[16] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[17] M. Bux and U. Leser. Parallelization in scientific workflow management systems. *arXiv preprint arXiv:1303.7195*, 2013.

[18] Y. Chen, T. Souaiaia, and T. Chen. Perm: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics*, 25(19):2514–2521, 2009.

[19] W. Cheng, M. Rademaker, B. De Baets, and E. Hüllermeier. Predicting partial orders: Ranking with abstention. In *Machine Learning and Knowledge Discovery in Databases*, pages 215–230. Springer, 2010.

[20] S. Cohen-Boulakia, A. Denise, and S. Hamel. Using medians to generate consensus rankings for biological data. In *Scientific and Statistical Database Management*, pages 73–90. Springer, 2011.

[21] S. Cohen-Boulakia and U. Leser. Search, adapt, and reuse: The future of scientific workflow management systems. *SIGMOD Record*, 40(2):6–16, 2011.

[22] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[23] D. Cook and L. Holder. Graph-based data mining. *Intelligent Systems*, 2000.

[24] J. C. Corrales, D. Grigori, and M. Bouzeghoub. Bpel processes matchmaking for service discovery. In *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, pages 237–254. Springer, 2006.

[25] F. Costa, D. de Oliveira, E. Ogasawara, A. A. Lima, and M. Mattoso. Athena: text mining based discovery of scientific workflows in disperse repositories. In *Resource Discovery*, pages 104–121. Springer, 2012.

[26] S. B. Davidson, S. C. Boulakia, A. Eyal, B. Ludäscher, T. M. McPhillips, S. Bowers, M. K. Anand, and J. Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4):44–50, 2007.

[27] S. B. Davidson, X. Huang, J. Stoyanovich, and X. Yuan. Search and result presentation in scientific workflow repositories. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, page 17. ACM, 2013.

[28] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[29] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, et al. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1(1):25–39, 2003.

[30] E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[31] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[32] R. Dijkman, M. Dumas, B. V. Dongen, R. Käärik, and J. Mendling. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36(2):498–516, 2010.

[33] R. Dijkman, M. Dumas, and L. García-Bañuelos. Graph matching algorithms for business process model similarity search. *Business Process Management*, pages 48–63, 2009.

[34] M. Dumas, L. García-Bañuelos, and R. Dijkman. Similarity search of business process models. *Data Engineering Bulletin*, 32(3):23–28, 2009.

[35] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001.

[36] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling. *Patterns: service-oriented architecture and web services*. IBM Corporation, International Technical Support Organization, 2004.

[37] T. Erl. *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.

[38] J. Freire, C. T. Silva, S. P. Callahan, E. Santos, C. E. Scheidegger, and H. T. Vo. Managing rapidly-evolving scientific workflows. In *Provenance and Annotation of Data*, pages 10–18. Springer, 2006.

[39] N. Friesen and S. Rüping. Workflow analysis using graph kernels. In *Proceedings of the ECML/PKDD Workshop on Third-Generation Data Mining: Towards Service-Oriented Knowledge Discovery (SoKD 2010), Barcelona, Spain*, 2010.

[40] D. Garijo, O. Corcho, Y. Gil, B. A. Gutman, I. D. Dinov, P. Thompson, and A. W. Toga. Fragflow automated fragment detection in scientific workflows. In *IEEE 10th International Conference on e-Science (e-Science), 2014*, volume 1, pages 281–289. IEEE, 2014.

[41] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *Ieee computer*, 40(12):26–34, 2007.

[42] Y. Gil, J. Kim, G. Florez, V. Ratnakar, and P. A. González-Calero. Workflow matching using semantic metadata. In *Proceedings of the fifth international conference on Knowledge capture*, pages 121–128. ACM, 2009.

[43] A. Goderis, P. Li, and C. Goble. Workflow discovery: the problem, a case study from e-science and a graph-based solution. In *International Conference on Web Services (ICWS) 2006*, pages 312–319. IEEE, 2006.

[44] A. Goderis, U. Sattler, P. Lord, and C. Goble. Seven bottlenecks to workflow reuse and repurposing. In *The Semantic Web–ISWC 2005*, pages 323–337. Springer, 2005.

[45] J. Goecks, A. Nekrutenko, and J. Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, 2010.

[46] O. Gospodnetic and E. Hatcher. *Lucene*. Manning, 2005.

[47] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, et al. Cybershake: A physics-based seismic hazard model for southern california. *Pure and Applied Geophysics*, 168(3-4):367–381, 2011.

[48] A. J. Hey, S. Tansley, K. M. Tolle, et al. The fourth paradigm: data-intensive scientific discovery. 2009.

[49] T. D. Huynh and L. Moreau. Provstore: a public provenance repository. *5th International Provenance and Annotation Workshop (IPAW'14)*, 2014.

[50] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 59–72. ACM, 2007.

[51] J.-Y. Jung, J. Bae, and L. Liu. Hierarchical business process clustering. In *IEEE International Conference on Services Computing, 2008. SCC'08.*, volume 2, pages 613–616. IEEE, 2008.

[52] G. Juve and E. Deelman. Scientific workflows and clouds. *Crossroads*, 16(3):14–18, Mar. 2010.

[53] M. Kendall. A new measure for rank correlation. *Biometrika*, 1938.

[54] J. Krinke. Identifying similar code with program dependence graphs. In *Proceedings of the 8th Working Conference on Reverse Engineering*, pages 301–309. IEEE, 2001.

[55] B. Langmead, C. Trapnell, M. Pop, S. L. Salzberg, et al. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[56] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady*, 10:707, 1966.

[57] H. Li and R. Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[58] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 1932.

[59] B. Ludascher, I. Altintas, and A. Gupta. Compiling abstract scientific workflows into web service workflows. In *15th International Conference on Scientific and Statistical Database Management*, pages 251–254, 2003.

[60] B. Ludäscher, K. Lin, S. Bowers, E. Jaeger-Frank, B. Brodaric, and C. Baru. Managing scientific data: from data integration to scientific workflows. *GSA Special Paper 397, Geoinformatics: From Data to Knowledge, ed. A. Krishna Sinha*, 2006.

[61] B. Ludäscher, M. Weske, T. McPhillips, and S. Bowers. Scientific workflows: Business as usual? *Business Process Management*, pages 31–47, 2009.

[62] R. Maclin and D. Opitz. Popular ensemble methods: An empirical study. *arXiv preprint arXiv:1106.0257*, 2011.

[63] F. Malucelli, T. Ottmann, and D. Pretolani. Efficient labelling algorithms for the maximum noncrossing matching problem. *Discrete Applied Mathematics*, 47(2):175–179, 1993.

[64] P. Mates, E. Santos, J. Freire, and C. T. Silva. Crowdlabs: Social analysis and visualization for the sciences. In *23th International Conference on Scientific and Statistical Database Management*, pages 555–564. Springer, 2011.

[65] F. McSherry and M. Najork. Computing information retrieval performance measures efficiently in the presence of tied scores. *Advances in Information Retrieval*, 2008.

[66] B. T. Messmer and H. Bunke. Efficient subgraph isomorphism detection: a decomposition approach. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):307–323, 2000.

[67] P. Missier, B. Ludäscher, S. Dey, M. Wang, T. McPhillips, S. Bowers, M. Agun, and I. Altintas. Golden trail: Retrieving the data history that matters from a comprehensive provenance repository. *International Journal of Digital Curation*, 7(1):139–150, 2012.

[68] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, R. Greenwood, K. Carver, M. Pocock, A. Wipat, and L. P. Taverna: a tool for the composition and enactment of bioinformatics workflow. *Bioinformatics*, 20(17):3045–3054, 2003.

[69] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110. ACM, 2008.

[70] M. Pruett. *Yahoo! pipes*. O'Reilly, 2007.

[71] U. Radetzki, U. Leser, S. Schulze-Rauschenbach, J. Zimmermann, J. Lüssem, T. Bode, and A. B. Cremers. Adapters, shims, and glue - service interoperability for in silico experiments. *Bioinformatics*, 22(9):1137–1143, 2006.

[72] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[73] P. Romano, E. Bartocci, G. Bertolini, F. De Paoli, D. Marra, G. Mauri, E. Merelli, and L. Milanesi. Biowep: a workflow enactment portal for bioinformatics applications. *BMC bioinformatics*, 8(Suppl 1):S19, 2007.

[74] D. Roure, C. Goble, and R. Stevens. The design and realisation of the myExperiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems*, 25(5):561–567, 2009.

[75] G. Salton and M. J. McGill. Introduction to modern information retrieval. 1983.

[76] E. Santos, L. Lins, J. P. Ahrens, J. Freire, and C. T. Silva. A first study on clustering collections of workflow graphs. In *Provenance and Annotation of Data and Processes*, pages 160–173. Springer, 2008.

[77] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and re-using workflows with vstrails. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1251–1254. ACM, 2008.

[78] V. Silva, F. Chirigati, K. Maia, E. Ogasawara, D. Oliveira, V. Braganholo, L. Murta, and M. Mattoso. Similarity-based workflow clustering. *Journal of Computational Interdisciplinary Sciences*, 2(1):23–35, 2011.

[79] J. Starlinger, B. Brancotte, S. Cohen-Boulakia, and U. Leser. Similarity Search for Scientific Workflows. *PVLDB*, 7(12), 2014.

[80] J. Starlinger, S. Cohen-Boulakia, S. Khanna, S. Davidson, U. Leser, et al. Layer decomposition: An effective structure-based approach for scientific workflow similarity. *IEEE 10th International Conference on e-Science*, 2014.

[81] J. Starlinger, S. Cohen-Boulakia, and U. Leser. (Re)Use in Public Scientific Workflow Repositories. *Scientific and Statistical Database Management*, pages 361–378, 2012.

[82] J. Stoyanovich, B. Taskar, and S. Davidson. Exploring repositories of scientific workflows. *WANDS*, pages 7:1–7:10, 2010.

[83] W. Tan, J. Zhang, and I. Foster. Network analysis of scientific workflows: a gateway to reuse. *IEEE Computer*, page 54, 2010.

[84] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[85] W. M. Van Der Aalst, A. H. Ter Hofstede, and M. Weske. Business process management: A survey. In *Business process management*, pages 1–12. Springer, 2003.

[86] S. Wandelt, A. Rheinländer, M. Bux, L. Thalheim, B. Haldemann, and U. Leser. Data management challenges in next generation sequencing. *Datenbank-Spektrum*, 12(3):161–171, 2012.

[87] I. Wassink, P. Vet, K. Wolstencroft, P. Neerincx, M. Roos, H. Rauwerda, and B. T.M. Analysing scientific workflows: Why workflows not only connect web services. *World Conference on Services*, pages 314–321, 2009.

[88] X. Xiang and G. Madey. Improving the reuse of scientific workflows and their by-products. *IEEE International Conference on Web Services*, pages 792–799, 2007.

[89] Z. Yan, R. Dijkman, and P. Grefen. Fast business process similarity search with feature-based similarity estimation. In *On the Move to Meaningful Internet Systems: OTM 2010*, pages 60–77. Springer, 2010.

[90] U. Yildiz, A. Guabtni, and A. Ngu. Business versus scientific workflows: A comparative study. *World Conference on Services*, pages 340–343, 2009.

[91] J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3(3-4):171–200, 2005.

[92] J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri. Recommend-as-you-go: a novel approach supporting services-oriented scientific workflow reuse. In *IEEE International Conference on Services Computing (SCC) 2011*, pages 48–55. IEEE, 2011.

[93] G. K. Zipf. The psycho-biology of language. 1935.

List of Figures

List of Tables

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Berlin, den 16.12.2014

Johannes Starlinger