Data Integration in the Life Sciences: Scientific Workflows, Provenance, and Ranking

Sarah Cohen-Boulakia

Data Integration in the Life Sciences: Scientific Workflows, Provenance, and Ranking

Habilitation à Diriger des Recherches

(Spécialité Informatique)

Université Paris-Sud

présentée et soutenue publiquement le 17 Juin 2015

par

Sarah Cohen-Boulakia

Jury

Rapporteurs:	Peter Buneman	Professeur, Université d'Edimbourg, Royaume Uni
	Val Tannen	Professeur, Université de Pennsylvanie, Etats-Unis
	Alain Viari	Directeur de Recherche, Inria, Grenoble, France
Examinateurs:	Christine Froidevaux	Professeur, Université Paris Sud, France
	Olivier Gascuel	Directeur de Recherche, CNRS, Montpellier France
	Ioana Manolescu	Directeur de Recherche, Inria, Saclay, France
	Patrick Valduriez	Directeur de Recherche, Inria, Montpellier, France

Laboratoire de Recherche en Informatique – CNRS UMR 8623
Université Paris Sud
Bâtiment Ada Lovelace 650, F-91405, Orsay Cedex France
Secrétariat : Tél.: +33 (0)1 69 15 66 29 – Fax : +33 (0)1 69 15 65 79 70
http://www.lri.fr

Mis en page avec la classe thloria.

CONTENTS

Sommaire

CHAPTER 1

INTRODUCTION

Main Publications of the Chapter

[CBL11a] **Next generation data integration for Life Sciences.**, ICDE 2011 (3 hours Tutorial)

[CBL11b] **Search, Adapt, and Reuse: the Future of Scientific Workflows.**, SIGMOD Record 2011

Biological research is a science which derives its findings from the proper analysis of experiments. Today, a large variety of experiments are carried-out in hundreds of labs around the world, and their results are reported in a myriad of different databases, web-sites, publications etc., using different formats, conventions, and schemas.

Providing a uniform access to these diverse and distributed databases is the aim of *data integration* solutions, which have been designed and implemented within the bioinformatics community for more than 20 years. The goals and benefits of data integration from a biologists point-of-view are numerous and include cost reduction (less redundant work to be performed implies many more reproducible experiments), quality enhancement (exploiting redundant work implies augmenting the confidence a user may have on a piece of data), new findings (combining complementary work), and faster discoveries (reusing instead of redoing). From the very beginning, database researchers have also worked on this topic, attracted by the importance and challenging aspect of the problem and the multitude of truly heterogeneous and freely available data sets.

The need for data integration in the life sciences has not ceased, and is ever increasing [GS08, Sea05]. But what has changed dramatically over the last three decades is the throughput of the experiments – from single observations to gigabytes of sequences in a single day – and the breadth of questions that are studied – from single molecules to entire genomes, transcriptomes, proteomes, etc. – placing the problem of combining and managing biological data directly within the context of *big data* integration.

Systems Biology, aiming at a comprehensive view on cell physiology, inherently depends on data from a multitude of different sources. Translational Medicine targets the transfer of results from basic biological research into medical practice, calling for the integration of genomic and medical data. Related disciplines such as ecology, paleontology, or biodiversity, increasingly need to include data on a level of detail that is only achieved by genomic research, often integrating biomolecular data with geographic information. This trend is also reflected in the establishment of proper workshops and conferences series (e.g., DILS - Data Integration in the Life Sciences), large project calls and dedicated institutes on national level (Institut de Biologie Computationnelle (Montpellier), Institut Francais de Bioinformatique,...) and international level (eScience, cyberinfrastructure, and the establishment of specialized working groups at international organizations such as the W3C Semantic Web Health Care and Life Sciences Interest Group...).

However, the perception of the problem of data integration research in the life sciences has changed: While early approaches concentrated on handling schema-dependent queries over heterogeneous and distributed databases, current research emphasizes instances rather than schemas, tries to place the human back into the loop, and intertwines data integration and data analysis. Transparency – providing users with the illusion that they are using a centralized database and thus completely hiding the original databases – was one of the main goals of federated databases. It is not a target anymore. Instead, users want to know exactly which data from which source was used in which way in studies (Provenance). The old model of "first integrate, then analyze" is replaced by a new, process-oriented paradigm: "integration is analysis - and analysis is integration".

This paradigm change gives rise to some important research trends. First, the process of integration itself, i.e., the *integration workflow*, is becoming a research topic in its own. Scientific workflows actually implement the paradigm "integration is analysis". A second trend is the growing importance of sensible *ranking*, because data sets grow and grow and it becomes increasingly difficult for the biologist user to distinguish relevant data from large and noisy data sets. A last important trend in the bioinformatics community is the use of semantic Web techniques to integrate biological data.

This HDR thesis outlines my contributions to the field of data integration in the life sciences. More precisely, my work takes place in the first two contexts mentioned above, namely, *scientific workflows* and *biological data ranking*. The reported results were obtained from 2005 to late 2014, first as a postdoctoral fellow in the Database group and the Penn Center for Bioinformatics (Dec 2005 to Aug 2007) and then as an Associate Professor at Université Paris-Sud (LRI, UMR CNRS 8623, Bioinformactics team) and Inria (Saclay-Ile-de-France, AMIB team 2009-2014). During this period, I have mainly worked on the problems of querying, comparing and sharing scientific workflows with a specific interest in provenance aspects, and the problem of designing consensus ranking approaches for biological data. Three PhD students have been involved in this work under my co-supervision. My work has been also done in the context of close collaborations with the Humboldt University of Berlin, University of Manchester, University of Pennsylvania and University of Montréal as well as with physicians from the Institut Curie and Children's Hospital of Philadelphia and with biologists (workflow users) from several institutes.

Sections A and B present the two lines of research mentioned above. These sections outline the content of the following chapters, namely, Chapter 2 (*Managing and Querying workflow Provenance*), Chapter 3 (*Workflow similarity and rewriting approaches to enhance workflow reuse*), and Chapter 4 (*Ranking biological data*). Finally, Chapter 5 outlines perspectives for future work.

A Scientific Workflows

1 Motivations

Typical analysis processes in the life sciences are complex, multi-staged, and large. One of the most important challenges is to properly represent, manage, and execute such in-silico experiments. As a response to these needs, scientific workflow management systems have been introduced. They provide an environment to guide a scientific analysis process from its design to its execution. This area is largely driven by the bioinformatics community and also attracts attention in fields like geophysics or climate research. In a scientific workflow, the analysis processes are represented at a high level of abstraction which enhances flexibility, reuse, and modularity while allowing for optimization, parallelization, logging, debugging etc. [WPF05]. Scientific workflow systems may also deal with failure handling, scheduling, and monitoring. In such systems all steps plus intermediate results are traced, making it possible to repeat and reproduce experiments.

Differences with business and ETL (Extract-Transform-Load) workflows have been studied extensively [YGN09a, Slo07, AMA06, SKD10, LWMB09, MBZL09]: scientific workflows have building blocks which are complex user-defined functions rather than relational operators and they are focused on data transformations.

These developments, accompanied by the growing availability of analytical tools wrapped as (web) services, were driven by a series of expectancies [CBL11b]: End users of scientific workflow systems, without any programming skills, are empowered to develop their own pipelines; reuse of services is enhanced by easier integration into custom workflows; time necessary for developing analysis pipelines decrease; etc. However, despite all efforts, scientific workflow systems have not yet found widespread acceptance in their intended audience.

In the meantime, it becomes possible to share, search, and compare scientific workflows, opening the door to the exchange of mature and specialized data integration solutions. For example, myExperiment [RGS09] is a portal that hosts more than two thousands scientific workflows while BioCatalogue [BTN$^+$10] is a repository of more than one thousand web services to be called and combined in workflows.

We argue that a wider adoption of scientific workflow systems would be highly beneficial for users but can only be achieved if at least the following three points are considered.

First, provenance in scientific workflows [DF08] is a key concept and should be considered as a first citizen in scientific workflow systems. The importance of replication and reproducibility has been critically exemplified [SNTH13] through studies showing that scientific papers commonly leave out experimental details essential for reproduction, studies showing difficulties with replicating published experimental results, an increase in retracted papers, and through a high number of failing clinical trials. Provenance supports reproducibility and allows assessing the quality of results. Research questions for workflow provenance include comparing workflow runs based on their provenance data and querying provenance information which can be, in turn, used to asses the similarity of workflows.

Second, since the targeted users are mainly non programmers, they may not want to design workflows from scratch. The focus of research should thus be placed on searching, adapting, and reusing existing workflows. Only by this shift can scientific workflow systems outreach to the mass of domain scientists actually performing scientific analysis - and with little interest in developing them themselves. To this end, scientific workflow systems need to be combined with community-wide workflow repositories allowing users to find solutions for their scientific needs (coded as workflows).

Third, in the same spirit, workflows should remain simple to use: a complex workflow composed of dozens of intertwined tasks, in general, is not much easier to understand than a well structured program performing the same analysis.

Our contributions to these research questions are depicted in the next two chapters of this HDR thesis and summarized here after.

2 Managing and Querying workflow Provenance

Chapter 2 addresses the problems of managing and querying provenance in scientific workflows and provides solutions designed in collaboration with biologists from the Ppod (Processing Phyl-O'Data) project.

This Chapter first introduces ZOOM*userview [BCBDH08] which addresses the problem of reducing the large amount of provenance information produced by workflow runs by providing abstraction mechanisms to focus on the most relevant information. Since bioinformatics tasks may themselves be complex sub-workflows, a user view determines what level of sub-workflow the user can see, and thus what data and tasks are visible in provenance queries. More specifically, we formalize the notion of user views [CCBD06], demonstrate how they can be used in provenance queries, and provide an algorithm for generating a user view based on the tasks considered as relevant for the user. We show the ability of ZOOM to manage the workflow of the first Provenance Challenge [CBBCD08]. We then describe our prototype [BCBD07].

The second part of the Chapter is devoted to the presentation of the PDiffView approach [BCBD$^+$09]. Here, we consider the problem of differing the provenance of two data products produced by executions of the same specification. Although this problem is NP-hard for general workflow specifications, an analysis of real scientific workflows shows that in majority their specifications can be captured as series-parallel graphs overlaid with well-nested forking and looping. For this restriction, we present efficient, polynomial-time algorithms for differencing executions of the same specification and thereby understanding the difference in the provenance of their data products. We then describe the prototype of PDiffView built around our differencing algorithm [BCBDG09].

3 Scientific workflows Reuse

Chapter 3 addresses two problems directly related to low workflow reuse.

First, we present the results on a workflow reuse study we performed on the users of the myExperiment repository [SCBL12]. Our conclusions depict the various aspects of the problem of low workflow reuse.

Second, we summarize the results obtained from a deep and large comparative review of workflow similarity approaches [SBCBL14]. More specifically, in this work we (i) compare in isolation different approaches taken at each step of scientific workflow comparison, reporting on an number of unexpected findings, (ii) investigate how these can best be combined into aggregated measures, and (iii) make available a gold standard of over 2 000 similarity ratings contributed by 15 workflow experts on a corpus of 1 500 workflows and re-implementations of all methods we evaluated.

We then introduce a novel and intuitive workflow similarity measure that is based on layer decomposition [SCBK+14]. Layer decomposition accounts for the directed dataflow underlying scientific workflows, a feature which has not been adequately considered in previous methods.

Third, we introduce techniques to reduce the workflow structural complexity as another attempt to make scientific workflows easier to (re)use. In this context, we present two approaches: DistillFlow and SPFlow.

On the one hand, DistillFlow [CBCG+14] aims to remove the structural redundancy in workflows designed with Taverna, one of the major scientific workflow systems. More precisely, our contribution is fourfold (i) we identify a set of anti-patterns that contribute to the structural workflow complexity, (ii) we design a series of refactoring transformations to replace each anti-pattern by a new semantically-equivalent pattern with less redundancy and simplified structure, (iii) we introduce a distilling algorithm that takes in a Taverna workflow and produces a distilled semantically-equivalent Taverna workflow, (iv) we provide an implementation of our refactoring approach [CCBF+14] that we evaluate on both the major public repository of Taverna workflows (myexperiment) and on a private collection of workflows from the BioVel project, in which libraries of Taverna workflows for processing data in biodiversity research are being designed.

On the other hand, SPFlow [CBFC12b] reduces the inherent complexity of workflows but contrary to DistillFlow (which is based on the semantics of Taverna), SPFlow is agnostic to the workflow system and based on more theoretical graph properties. In particular, SPFlow proposes to rewrite workflow graphs into series-parallel structures, for which the problem of subgraph isomorphism is polynomial.

Contributions of this Chapter are associated with the implementation of systems, which are successfully used by scientific workflow systems users. In particular, DistillFlow is used several times each week by a regular set of users (who are regular Taverna's users).

B Ranking

1 Motivations

Another key challenge in data integration in the life sciences is to help users choose between alternative pieces of information, such as choose between conflicting updates when maintaining a datawarehouse [Ive09], choose between different data sources [CBDF+06], or choose among several answers when querying a data integration system [BY06]. Such choices are best supported by sensible ranking methods [BNL+04]. However, even widely used portals still do not provide any ranking services although queries often produce huge amounts of results. For instance, searching for the set of human genes involved in breast cancer returns 2,000+ answers in the reference database EntrezGene without any elaborate ranking.

Despite the large body of work on ranking search results performed in the database community [CGHX06], no approach is currently able to take into account features of life science data. Biological data entries often are text-centric, which requires the inclusion of text mining methods in the ranking [LRSS08]. Facts are not always of equal strength. In the example above, some genes may be clearly involved in breast cancer while for others this relationship with cancer might still be putative. Links between entries are very important to augment the confidence in results; however, they cannot be considered in isolation, but only in the context of the entire network of linked entities [CBDF+06]. This may be achieved using PageRank-style algorithms in systems such as Biozon [BY06] or Soft-rank [VHR+09] while various criteria (e.g., the reliability of sources providing data or links) should additionally be considered [CBLS+04]. End users also have different expectations, such as either looking for established or for more surprising results [HTL07].

2 Consensus ranking approaches

The originality of our approach in the domain of biological data ranking lies in considering *consensus rankings* approaches. Instead of combining ranking criteria and designing yet another ranking function, we propose to consider several ranking criteria and methods, and generate a consensus ranking to highlight the common points of a set of rankings while minimizing their disagreements. Our work is based on the concept of *median*, originally defined on permutations: Given m rankings (provided by m ranking methods) and a distance function, the median problem is to find a ranking that is the closest of the m given permutations.

Chapter 4 presents our contributions in the design and use of consensus ranking approaches. First, while the problem of computing a median of a set of m rankings is known to be NP-hard (for $m >= 4$, m even), we introduce BioConsert (Biological Consensus ranking with ties) [CBDH11], an heuristic able to compute a consensus ranking. BioConsert is able to consider rankings of different elements and involving ties (where elements are equally rated). Then, we describe the results obtained by developing ConQuR-Bio [BRDCB14] which applies consensus ranking techniques to rank alternative results to a scientific query posed to the NCBI (National Center for Bioinfomatics) databases (set of major biological databases). The Chapter ends with another example of use where consensus rankings are exploited to aggregate the ratings performed by experts on workflows similarity, showing the complementary of data ranking and scientific workflows domains.

The work depicted in this Chapter has been done in close collaboration with pediatricians and oncologists from the Institut Curie and the Children's Hospital of Philadelphia. ConQuR-Bio, designed based on their needs, is currently used by 50 regular users from 5 different institutes.

Sommaire

CHAPTER

2

MANAGING AND QUERYING PROVENANCE IN SCIENTIFIC WORKFLOWS

Main Publications of the Chapter

[DCBE+07] **Provenance in Scientific Workflow Systems**, IEEE Data Eng. Bull. 2007
[CBT09] **Provenance in Scientific Databases**, Encyclopedia of Database Systems 2009
[BCBD07] **Zoom*UserViews: Querying Relevant Provenance in Workflow Systems**, VLDB 2007 (Demonstration)
[CBBCD08] **Addressing the provenance challenge using ZOOM**, Concurrency and Computation: Practice and Experience 2007
[BCBDH08] **Querying and Managing Provenance through User Views in Scientific Workflows**, ICDE 2008
[BCBD+09] **Differencing Provenance in Scientific Workflows**, ICDE 2009
[BCBDG09] **PDiffView: Viewing the Difference in Provenance of Workflow Results**, VLDB 2009 (Demonstration)

The complex computational processes designed to analyze and understand scientific data can be represented as workflows composed of numerous tools, each with various parameters. When executed, each workflow may generate huge sets of final and intermediate data products. Keeping track of the workflow specification provenance (exact version of the tools used, order in which tools are combined...) is of paramount importance to make reproducible experiments. Tracing the provenance of each intermediate and final data item generated by a workflow execution is necessary to allow the user to fully understand the outcome of each experiment. Challenges in managing and querying provenance information in scientific workflows are numerous and include determining the level(s) of granularity at which provenance information should be recorded, storing and accessing huge amounts of provenance information, reducing the amount of provenance information to be inspected by users...

Database provenance and annotations have been studied extensively by the database community [CW01, BSHW06, BCTV04, BCC06, BKT01]. In such projects, the aim is to determine which tuples were used to generate the answer to a query by exploiting the algebraic form of the query and/or the relational or XML-like form of the data. A common framework based on semiring annotations has been proposed in [GKT07]. In contrast, transformations occurring in workflows are usually external processes (black boxes), and the log files typically provide mainly object ids. Provenance is more coarse-grained, and the structure of data cannot be reasoned about. Based on such salient differences with database provenance, the problematic of provenance in scientific workflows has been gaining interest since the early 2000s. Several projects have started to consider the problem of designing and exploiting provenance modules in scientific workflow systems: Taverna [MSRO+10], Kepler [BML+06], Chimera/Triana [FVWZ02], and VisTrails [FSC+06], to mention a few, have actively worked on this area.

At the University of Pennsylvania, the pPOD (*Processing Phyl-O'Data*) project was at the center of this problematic. The aim of pPOD was to provide methods and tools, assembled through pipelines, for data integration and interoperability to the series of ATOL projects (*Assembling the Tree Of Life*, series of 30+ projects in phylogeny). Reproducing experiments and understanding why a given pipeline output a given phylogenetic tree instead of another was at the heart of the project. Provenance management was thus of paramount importance.

In the meantime, a series of international *Provenance Challenges* [Mea08] has been organized. 50 international attendees grouped into 18 teams (from 14 different institutes) have participated to the challenge. The aim of such challenges was to provide a better understanding of the capability of the numerous workflow systems in terms of provenance management. A reference workflow was provided by the organizers and each participant to the challenge had to implement the workflow into his system, run several executions, store intermediate and final data results and then answer a set of provenance queries. The too large variety of systems involved did not allow to go to a full interoperability between any pair of systems. The design of the Open Provenance Model (OPM) [MFF+08] proposed as a common model for Provenance management was probably too generic to properly answer the concrete needs of provenance management in scientific workflow systems. However, such events had a major impact in the scientific workflow community. In particular it underlined the fact that provenance queries were neither equally difficult to be answered by the various teams, nor interpreted the same way by the participants.

More specifically, two points deserve attention.

First, the amount of provenance information generated by a workflow execution was large. This first point allowed us to demonstrate during the first provenance challenge the interest of using ZOOM*UserViews (ZOOM for short) to manage and query provenance. ZOOM is a provenance querying system we have designed and developed to answer needs expressed within the context of the pPOD project. This is the topic of Section A.

Second, one of the queries to be asked during the challenge was related to the comparison between two executions of the same workflow. Several participants were able to answer this query in very simple settings (where a single execution of module was impacted from one execution to the other) while they were not able to answer it in more general settings. This point, together with the need of comparing executions of pipelines met in pPOD, made us addressing the problem of differing executions of scientific workflows. This is the topic of Section B.

A Provenance through user views

1 Motivation

While provenance information should help users understand the results they obtained to a given experiment, the amount of provenance information provided to them may be so large that it cannot be exploited. There is thus a need to reduce the amount of provenance and help users first inspect the data of high relevance and then zoom into more details.

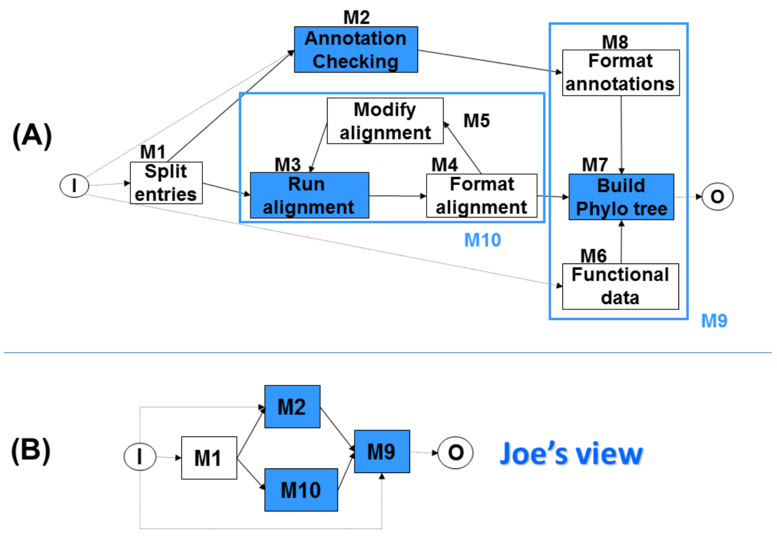


FIGURE 2.1 : Workflow and Joe's view

As an example, consider the workflow specification (a.k.a workflow definition or schema) in Figure 2.1 (A) (do not consider the big boxes and colors for now), which describes a common analysis in molecular biology: *Phylogenomic inference of protein biological function*, designed in the pPod project. This workflow first takes in a set of entries selected by the user from a database (such as GenBank), and formats these entries to extract a set of sequences, and, possibly, a set of annotations (M1). An alignment is then created (M3), and the result formatted (M4). The user may also be interested in rectifying the alignment (M5). M3 to M5 are repeated until the biologist is satisfied with the result obtained. The user may also inspect the annotations provided by GenBank (M2) and generate a set of curated annotations; new user input is needed for this. The annotations are then formatted (M8) to be taken as input to the phylogenetic tree reconstruction task (M7). Other annotations are also considered: M6 takes in annotations from the user's lab and formats them to be taken as input to M7. From the annotations produced by M8 (and possibly M6) together with the alignment produced by M4, M7 provides a phylogenetic tree labeled with functional annotations. Note that a number of these tasks or *modules* (e.g. M1, M4, M8) involve formatting and are not central to the scientific goal of the experiment, and that edges represent the precedence and potential *dataflow* between modules during an execution.

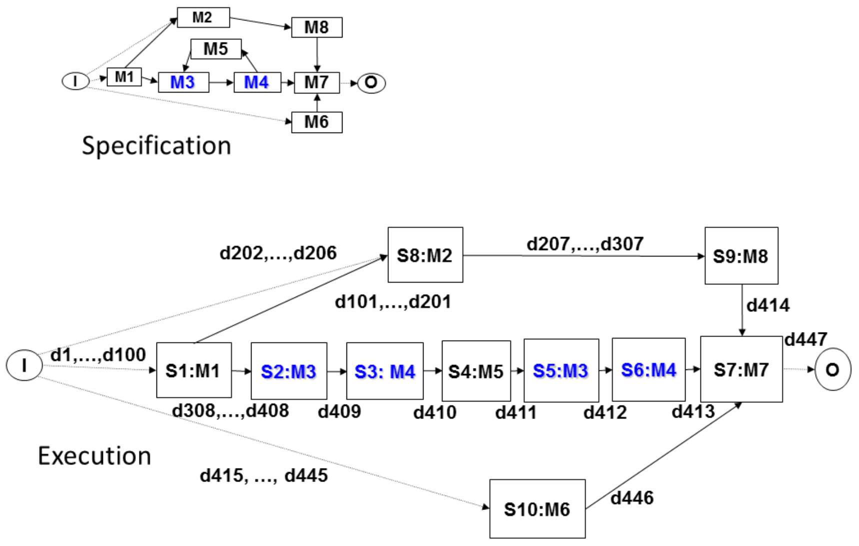


FIGURE 2.2 : Example of run

Workflows may be executed several times, resulting in vast amounts of intermediate and final data objects. Figure 2.2 shows an execution of the workflow in Figure 2.1 (A). In an execution, each box (e.g. S1:M1) is called a *step* and represents the execution of a module; in the figure, it is labeled both with a unique id for the step (e.g. S1) as well as the module of which it is an execution (e.g. M1). Edges in this execution are labeled with data ids (e.g. d1, d202, d447) representing the actual data that is used/created during the execution. In the workflow execution of Figure 2.2, one hundred sequences are taken as initial input (d1 to d100), minor modifications are done on the annotations (d202 to d206), and thirty additional annotations are used (d415 to d445).

As stated above, since a workflow execution may contain many steps and data objects, the amount of provenance information can be overwhelming. For example, the provenance of the final data object d447 in Figure 2.2 would include every data object (d1,...,d447) and every step (S1,...,S10). There is therefore a need for abstraction mechanisms to present the most *relevant* provenance information to the user.

Intuitively, we want to allow users to group modules together to get a workflow in which composite modules represent some relevant task. For example, suppose user Joe believes “Annotations checking" (M2), “Run alignment” (M3) and “Build Phylo tree” (M7) (shaded boxes in Figure 2.1) to be relevant. Then he might group M6, M7 and M8 together in composite module M9 (shown as a dotted box), which then takes on the meaning of relevant module M7, e.g. building a phylogenetic tree. Similarly, he might group M3, M4 and M5 together in composite module M10, which takes on the meaning “Run alignment”.

However, users may differ in their interests: for example, while Joe is not interested in the alignment modification step (M5) (see Figure 2.1 where this step is not colored), another user, Mary, may be (see Figure 2.3 (A) where this step is colored). Mary would therefore not include M5 in her composite module

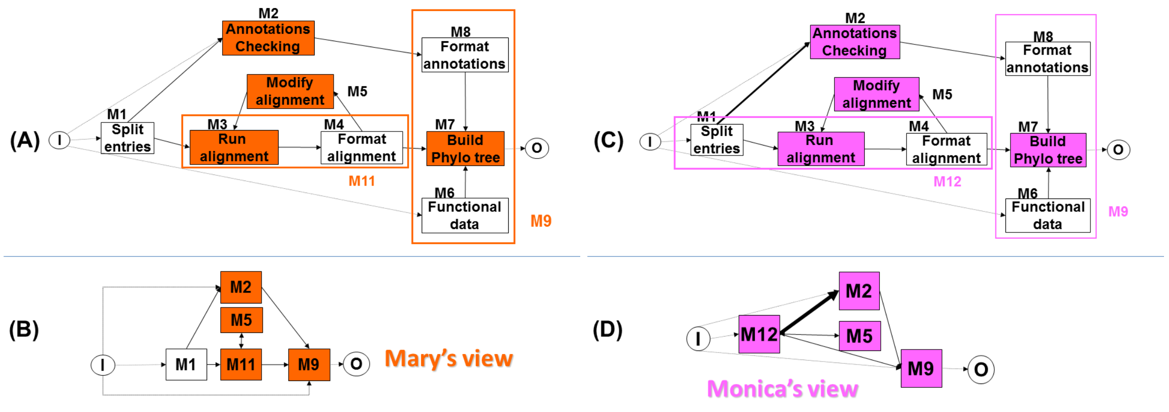


FIGURE 2.3 : Two alternative user views.

representing "Run alignment"; M11 includes only M3 and M4, leaving M5 visible. She may, however, agree on composite module M9. Joe and Mary will therefore have different *user views* defining the level of granularity at which they wish to view the workflow. Using the composite modules in the user view, an *induced* workflow, corresponding to the user view of Mary, is created (see Figure 2.3 (B)).

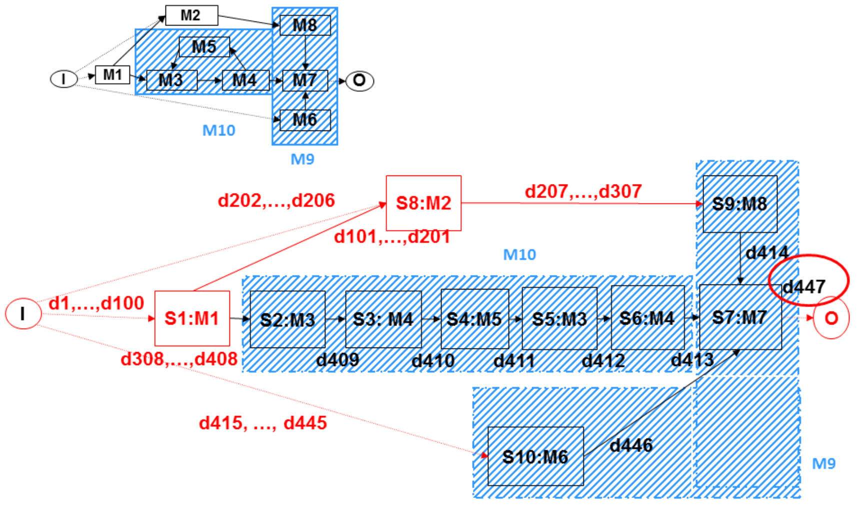


FIGURE 2.4 : (a) Workflow with composite modules; (b) Workflow run with composite executions

Now, let us consider a third user, Monica. Since M1 is not relevant to Monica, she may now wish to group it with M3 and M4 as depicted in Figure 2.3 (C). However, these groupings would dramatically modify the dataflow between relevant modules she perceives. For example, by grouping M1 with M3 and M4 in a new composite module M12, there exist an edge from M12 to M2 in the view (see Figure 2.3 (D)), due to the edge from M1 to M2 in the workflow specification. That is, it appears that "Run alignment" (the relevant module of M12) must be performed before "Run alignment" (M3), when in fact there is no precedence or dataflow between those modules. We must therefore restrict what groupings can occur so as to preserve the precedence between relevant modules and hence the perceived data provenance.

2 Contributions

Our contribution is threefold: we introduce a formalization of the concepts of user view and composite modules; we state the properties that composite modules should respect for user views to be safe and complete and we provide a polynomial-time algorithm able to compute automatically user views based on the tasks considered as being of interest for the user; we provide a prototype for ZOOM able to compute user views on workflows and project views on runs to lower the amount of provenance provided to the user. Details are available in the corresponding publications [BCBDH08, CBBCD08, BCBD07] while the main points are provided here after.

2.1 Formalizing and constructing user views

We first introduce the concept of workflow specification on which user views are based.

Workflow specification. A *workflow specification* defines the order in which modules can be executed and indicates dataflow. More formally, it is a directed graph, $G_w(N, E)$, in which nodes are uniquely labeled modules. Two special nodes, *input* (I) and *output* (O), are source and sink nodes, respectively, and indicate the beginning and end of the workflow. Every node of G_w must be on some path from *input* to *output*.

User view. A *user view* U of a workflow specification is a partition of its nodes N (excluding *input* and *output*), that is a set $\{M_1, ..., M_n\}$ such that $\emptyset \neq M_i \subseteq N$, M_i and M_j are disjoint for $i \neq j$, and $M_1 \cup M_2 \cup ... \cup M_n = N$. Each M_i is called a *composite* module. The *size* of U, $|U|$, is the number of composite modules it contains. For example, the size of Joe's user view is 4 while that of Mary is 5.

A user view $U = \{M_1, ..., M_n\}$ of a workflow specification G_w *induces* a "higher level" workflow specification, $U(G_w)$, in which there is a node for each M_i (labeled with a new composite module name), *input* and *output* nodes, and an edge $M_i \longrightarrow M_j$ whenever there is an edge in G_w between a module in M_i and a module in M_j (similarly for edges $input \longrightarrow M_i$ and $M_i \longrightarrow output$). The induced specification for Joe's and Mary's user views are shown in Figure 2.1 (B) and 2.3 (B), respectively.

Workflow run. An execution of a workflow specification is called a *workflow run*. It generates a partial order of *steps*, each of which has a set of *input* and *output* data objects. More formally, it is a directed acyclic graph, G_r, in which nodes are labeled with unique step-ids as well as the modules of which they are executions. Module labels are not necessarily unique due to cycles in the workflow specification, that is, loops in the specification are unrolled in the execution. For example, in Figure 2.2 there are two executions of M3, S2 and S5, since the loop between M3 and M5 was executed twice. Edges are labeled with a unique edge label indicating the ids of the data output by the source step and input to the target step. Nodes *input* and *output* indicate the beginning and end of the execution, respectively; every node must be on some path from *input* to *output*.

Provenance. Each data object in the workflow dataspace is produced either as a result of a step of a workflow run, or is input by the user. We call the *provenance* of a data object the sequence of modules and input data objects on which it depends [Mor06, CCBD06]. If the data is a parameter or was input to the workflow execution by a user, its provenance is whatever metadata information is recorded, e.g. who input the data and the time at which the input occurred. Following other work (e.g., [BSHW06, BL05]), we assume data is never overwritten or updated in place. Each data object therefore has a unique identifier and is produced by at most one step.

We assume that each workflow run generates a log of events, which tells what module a step is an instance of, what data objects and parameters were input to that step, and what data objects were output from that step. For example, the log could include the start time of each step, the module of which it was an instance, and read (write) events that indicate which step read (wrote) which data objects at what time.

Using this log information, we can determine the *immediate provenance* for a data object as the step which produced it and the input set of data objects. The *deep provenance* for a data object is recursively defined as all the steps and input set of data objects that were transitively used to produce it.

For example, the immediate provenance of the data object d413 in the workflow run of Figure 2.2 is the step with id S6, which is an instance of the module M4, and its input set of data objects {d412}. The deep provenance of d413 includes all the steps and their inputs that transitively produced it, and would include (among other things) the step with id S2, which is an instance of the module M3, and its input set of data objects {d308,...,d408}.

Composite executions. The execution of consecutive steps within the same composite module causes a virtual execution of the composite step, shown in Figure 2.2 by dotted boxes. These virtual executions can be constructed from the log information as well as containment information between modules and composite modules. For example, we would construct an execution of M10 with the id S13 in Figure 2.2, which takes as input the set of data objects {d308,...,d408} and produces as output {d413}. Similarly, we would construct two executions of M11, the first of which (id S11) takes as input {d308,...,d408} and produces as output {d410}, and the second of which (id S12) takes as input {d411} and produces {d413} (see [CBBCD08] for details).

Since user views are defined in terms of composite modules, they restrict what provenance information can be seen in an execution by hiding internal steps as well as the data passed between internal steps. For example, the immediate provenance of d413 seen by Joe would be S13 and its input, {d308,...,d408}, since composite module M10 is in his user view, whereas that seen by Mary would be S12 and its input, {d411}, since M11 is in her user view. The deep provenance of d413 as seen by Mary would include the first execution of M11, S11, and its input {d308,...,d408}. However, Joe would not see the data d411, nor would he be aware of the looping inside of S13, i.e. the two executions of M3.

2.2 The ZOOM Algorithm

The aim of ZOOM is to construct user views using a bottom-up approach. Such a process takes as input a workflow specification and a set of relevant modules, and produce as output a user view. In this section, we provide the main properties that any "good" user view should satisfy.

From discussions with our scientist collaborators from the pPOD project, a user view is (intuitively) good if (i) the user sees a composite module for each relevant module. The composite module takes on the meaning of the relevant module it contains, hence in the induced workflow the paths (dataflow) between relevant modules (as represented by their composite modules) should be preserved. No path should be (ii) added or (iii) removed from the original workflow. However, the need to preserve paths between relevant modules may mean that it is impossible to include every non-relevant module in the specification in some composite module which represents a relevant module. That is, we may need to create one (or more) composite modules that do not contain a relevant module. Since such composite modules have no meaning in terms of a relevant module, (iv) there should be as few as possible.

Observations (i) to (iii) have been formalized by Properties 1 to 3 and (iv) by a minimality condition.

Property 1 *Given a workflow specification G_w and a set $R \subseteq N$ of relevant modules, a user view U is* well-formed *iff every composite module in U contains at most one element of R.*

Given a well-formed user view U for (G_w, R) and $n \in N$, we use $C(n)$ to denote the composite module in U which contains n. For simplicity, we extend the notation to include $C(input) = input$ and $C(output) = output$. Furthermore, we use the term `nr-path` to denote a path in G_w (or in view $U(G_w)$) which contains no relevant intermediate module $r \in R$ (or relevant composite module $C(r)$). As an example, in the workflow specification of Figure 2.1, there exists an `nr-path` from $input$ to M2, but not from $input$ to M7, since all paths connecting these two modules contain an intermediate node in R (M2, M3).

Property 2 *A user view U* preserves dataflow *iff every edge in G_w that induces an edge on an* `nr-path` *from $C(r)$ to $C(r')$ in $U(G_w)$ lies on an* `nr-path` *from r to r' in G_w. Here, r, r' are nodes in $R \cup \{input, output\}$.*

Property 3 *A user view U is* complete w.r.t dataflow *iff for every edge e on an* `nr-path` *from r to r' in G_w that induces an edge e' in $U(G_w)$, e' lies on an* `nr-path` *from $C(r)$ to $C(r')$. Here, r, r' are nodes in $R \cup \{input, output\}$.*

In other words, every `nr-path` from $C(r)$ to $C(r')$ in $U(G_w)$ must be the residue of an `nr-path` from r to r' in G_w, and each `nr-path` in G_w must have a residue in $U(G_w)$.

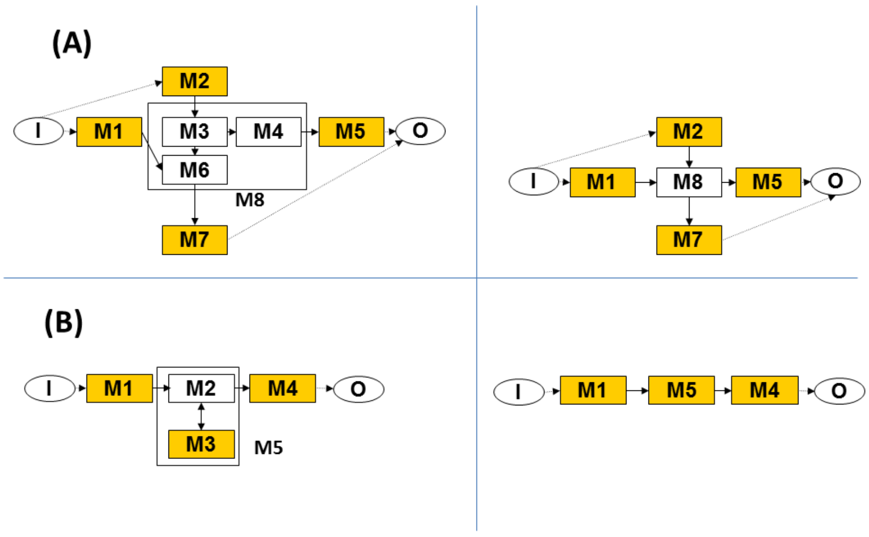


FIGURE 2.5 : Counter examples. Modules of interest for the user are colored.

As an example, consider the two workflows (left) and corresponding user views (right) shown in Figure 2.5. On part (A) of the figure, while the user view is well-formed, it does not preserve dataflow since it is possible to go from M1 to M5 in the user view while this is impossible in the original specification. This gives the impression that $M1$ produces data necessary for $M5$.

On part (B) of Figure 2.5, the user view is not complete w.r.t. dataflow since in the user view it is impossible to go from M1 to M4 without going through M5 which is not the case in the original specification.

We design a polynomial-time algorithm based on these three properties and on a minimality condition [BCBDH08]. The minimality condition ensures that it is not possible to merge any two composite modules produced by the algorithm and still get a valid solution. However, ZOOM does not always provide a solution which is minimum (there may be an alternative solution with a lower total number of composite modules). Biton *et al.* [BDKR09] have shown that when the workflow specification is a series-parallel graph [VTL79] then it is possible to design a polynomial-time algorithm for ZOOM able to produce minimum solutions.

2.3 ZOOM Prototype

We have implemented ZOOM to query the provenance information provided by a workflow system, as well as to help users construct an appropriate user view. The prototype is available at
`http://zoomuserviews.db.cis.upenn.edu/cgi-bin/pmwiki.php`.

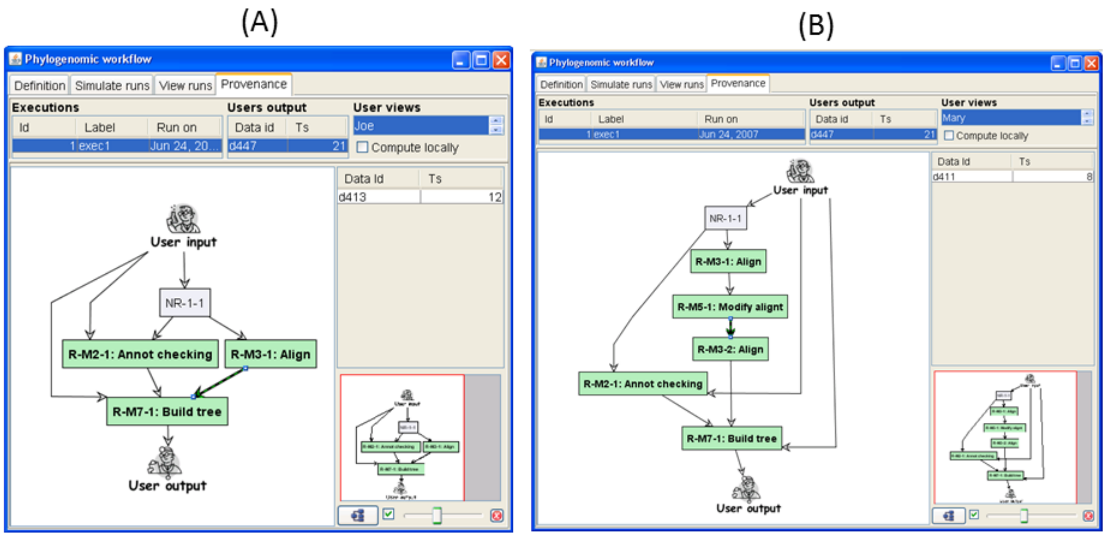


FIGURE 2.6 : Two screenshots of ZOOM: (A) Joe's and (B) Mary's user views.

In Figure 2.6 two screenshots are provided on the *Provenance* panel of the tool. The screenshots represent respectively the user view of Joe and Mary applied to the same workflow run. The execution of modules are prefixed by "R" (relevant, in green) and "NR" (non relevant, in grey) and by a number i corresponding to the ith execution of a given module. For example, in (B) "R-M3-2: Align" corresponds to the second execution of the relevant module named Align and identified by M3 in the specification.

In this example, the view of Mary (in Figure 2.6 (B)) is able to see two executions of the "Align" module and one execution of the "Modify alignt" module. Joe (in Figure 2.6 (A)) is only able to see one execution of the "Align" module while the "Modify alignt" module is completely hidden for him. By clicking on edges between boxes, the provenance information can be visualized: the data id $d413$ can be visualized in (A) as the output of the step "R-M3-1 Align" while $d411$, the output of "R-M5-1 : Modify Alignt" is only visible in (B).

B Differing executions of the same workflow

We now address the problem of differing workflow executions.

1 Motivation

As stated in the introduction of this Chapter, while most of the participating systems to the provenance challenge gave reasonable answers for simple workflow models [Mea08], the techniques used were not extendable to more complex execution models, i.e., supporting forked executions over an unknown number of elements of an input set (*implicit iteration*), looping until some condition is met (*explicit iteration*), or parallel executions.

As an example of a complex workflow, consider a scientific analysis involving protein annotation (See Fig. 2.7 (a)) inspired from a Kepler workflow designed within the pPOD project. The aim of this analysis is to infer the biological function of a new sequence from other sequences, either protein or domain (consecutive parts of proteins) sequences. The underlying biological assumption is that a protein's biological function is usually a composition of the biological functions of its domains. The main steps of the process are as follows: The user provides a sequence of protein (s) which is first converted into Fasta format (1) and compared using the BLAST tool to all the protein sequences of major proteomic resources[1], namely SwissProt (2), TrEMBL (3), and PIR (4). The most similar protein sequence (Top-1) found is selected (5) and can then be used to be blasted against SwissProt/PIR/TrEMBL sequences at its turn ("reciprocal best hits") and compare to the previous ones got until a set of very close proteins is found. The following steps are done for each sequence output of step (5), independently from each others. If proteomic domains are unknown, they are searched for in major domain resources such as ProDom (7) or PFAM (8); otherwise these steps may be skipped. Domain sequences are then extracted (9) and each domain is used as input to be annotated by ontologies (Gene Ontology (10) then FunCat (11)) or using enzymatic terms (Brenda (12) then Enzyme (13)). Annotations obtained are eventually gathered (t).

To represent such a workflow specification, we extend dataflows with a few control flow information for forks and loops. For example, the specification graph for our protein annotation example can be found in Fig. 2.7 (a). In this graph, a loop is indicated by a dotted backarrow, e.g., from module 5 (collectTop1&Compare) to module 1 (FastaFormat), and forking is indicated by a dotted oblong, e.g., the oblong around module 2 (BlastSwP) indicates that similar proteins can be searched for simultaneously. Note that this workflow could also be expressed using BPEL [bpe]. However, to simplify the presentation we will use a simpler notation that is also closer to what is used in most scientific workflow systems.

In a run, loops are unrolled and the number of fork executions is given explicitly. For example, two runs of the protein annotation workflow specification are shown in Fig. 2.7 (b) and (c). Observe that Run (b) has two fork executions between modules 5 and t, while Run (c) has two executions of the loop from module 5 to module 1.

In a dataflow execution, module names do not repeat and there is an immediate pairing between nodes in the two executions. Therefore, the naive approach of taking the difference of the nodes and edges in the two runs to calculate their difference works generally well. However, for the runs in Fig. 2.7 this approach does not work since node names repeat and hence there are many possible ways of pairing nodes. To determine the best pairing of nodes, a global computation must be performed to match copies

[1]SwissProt, TrEMBL and PIR are three subsections of UniProt, considered independently by users because they are associated to different annotation processes.

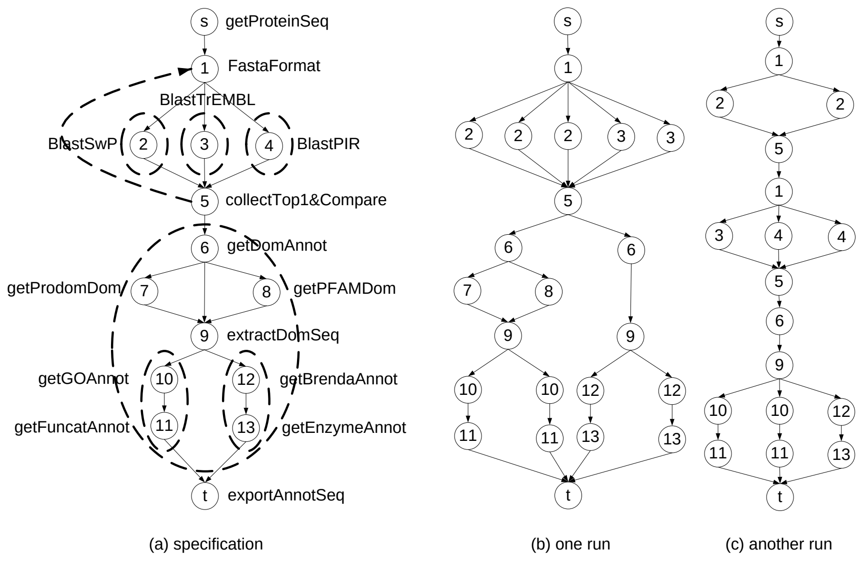


FIGURE 2.7 : Protein annotation workflow specification and runs

that are most similar overall in terms of the control structure and dataflow.

The difference or edit distance between a pair of valid runs of the same specification is defined as a minimum cost sequence of edit operations that transforms one run to the other. As many edit operations could be considered (e.g., insert or delete a node, and insert or delete an edge), it is important that they *transform a valid run to another valid run*, are *atomic*, and are *complete*. While inserting or deleting a node or edge are atomic operations that can be used to transform one valid run into another, they do not guarantee the validity of intermediate results. We therefore use as edit operations the insertion or deletion of *elementary paths*, that is, paths such that each internal node has exactly one incoming edge and one outgoing edge.

For example, an edit script from run (a) to run (b) would include (among other things) the insertion of the path (1,4,5).

This notion of edit distance has a simple appealing interpretation: It is the shortest path connecting the given pair of runs in the space of all valid runs, where two valid runs are adjacent iff they differ by a single elementary path.

While differing flow networks is NP-hard for general graphs [BCBD$^+$08], the structure of a majority of available workflows can be captured as a series-parallel graph (SP-graph), overlaid with well-nested forks and loops. For this restriction, we present efficient, polynomial-time algorithms for differencing workflow runs of the same specification. The algorithms are based on a well-known tree representation of SP-graphs in which internal nodes are annotated with *series* (in which case the children are ordered) or *parallel* (in which case the child nodes are unordered). We build on this by using additional annotations to represent loop (ordered) and fork (unordered) executions (*annotated SP-trees*).

2 Contributions

Our contributions are the following. First, we present a model of workflows that is more expressive than basic SP-graphs, allows to capture structural variations of runs while allowing less complex treatment than the general case. Second, for this model of workflows we design efficient, polynomial-time algorithms for differencing workflow executions, considering forks and loops. Our algorithms work under fairly general cost models, allowing us to capture a variety of application-specific notions of distance. Third, we implement the system **P**rovenance **Diff**erence **View**er (PDiffView) built around our differencing algorithm allowing users to see the difference in executions, and in particular to zoom in and out of the execution difference to see varying levels of detail.

2.1 SPFL workflows

SP graphs are a natural class of directed graphs. We provide here the definition of series-parallel graphs while we will go back to this definition and provide more examples in the following chapter.

Définition 2.1 A **series-parallel graph** (also called **SP-graph**) is a directed multigraph G with a single source s and a single sink t (two terminals) that can be produced by a sequence of the following operations:

- **Basic SP-graph:** Create a new graph consisting of a single edge directed from some node s to some node t.
- **Series Composition:** Given two SP-graphs G_1 and G_2 with sources s_1, s_2 and sinks t_1, t_2 respectively, form a new graph $G = S(G_1, G_2)$ by identifying $s = s_1$, $t_1 = s_2$ and $t = t_2$.
- **Parallel Composition:** Given two SP-graphs G_1 and G_2 with sources s_1, s_2 and sinks t_1, t_2 respectively, form a new graph $G = P(G_1, G_2)$ by identifying $s = s_1 = s_2$ and $t = t_1 = t_2$.

We can now define an Series-Parallel-Fork-Loop (SPFL) workflow specification by a triple $(G, \mathcal{F}, \mathcal{L})$, where G is a series-parallel graph with unique labels on the nodes, and $\mathcal{F}$ and $\mathcal{L}$ are two sets of subgraphs of G describing the well-nested set of allowed forks and loops respectively (such subgraphs are actually called Laminar graphs in the graph theory).

A workflow run R is then produced by applying a sequence of series, parallel, fork and loop executions recursively on the given specification. Intuitively, a series execution executes its sequential components in series; a parallel execution chooses a nonempty subset of all branches and executes them in parallel; a loop execution unfolds the cycle and executes all iterations of the loop in series; and a fork execution replicates one or more copies of the subgraph and executes them in parallel.

A complete formalization of the workflow specification, run, and the notion of run valid w.r.t. the specification is available in [BCBD^{+}09].

2.2 Difference Algorithm

In this context, the goal of differencing two runs is to find the minimum changes that transform the first run to the second. We consider four kinds of *path edit operations*: (1) *Path Insertion:* Create a new path between two existing nodes; (2) *Path Deletion:* Remove a path (inverse of path insertion); (3) *Path Expansion:* Create a new iteration of a loop by inserting a path between two existing consecutive iterations; and (4) *Path Contraction:* Remove an iteration of a loop by contracting the last path (inverse of path expansion). Note that edit paths must be *elementary* such that each internal node has exactly one incoming edge and one outgoing edge, and must transform one valid run to another valid run.

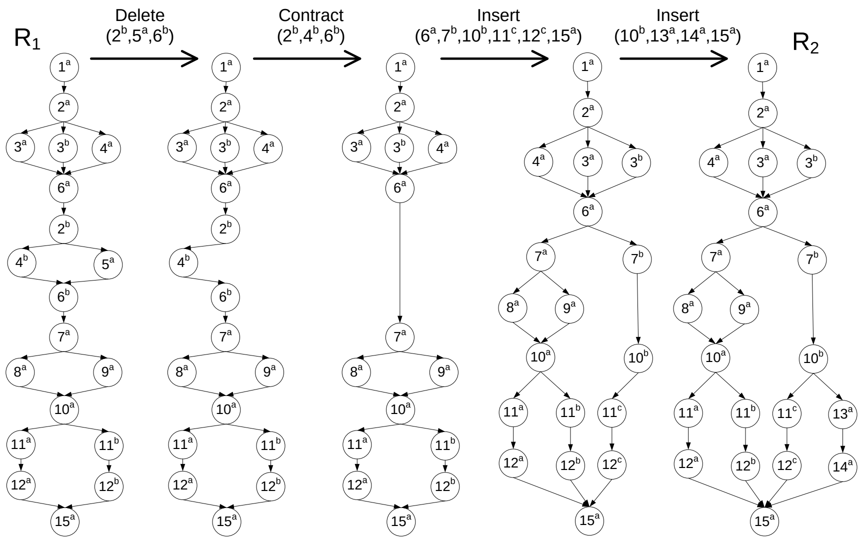


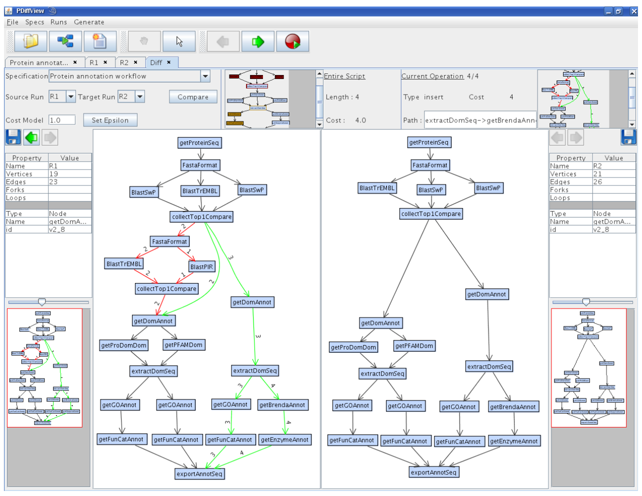
FIGURE 2.8 : A path edit script from R_1 to R_2

Exemple 2.1
Consider the runs R_1 and R_2 in Figure 2.7. A path edit script that transforms R_1 to R_2 is shown in Figure 2.8. Note that each intermediate run is valid with respect to the specification in Figure 2.7(a).

The polynomial-time algorithm computes the minimum-cost edit script [BCBD$^+$09] by reliying on a well-known tree representation of SP-graphs [VTL82] with extra annotations for forks and loops, and adopts a very general cost model: Each path edit operation is assigned a cost of l^ϵ, where ϵ is a user-specified real number no greater than one and l is the length of path to be edited. This feature allows users to capture a variety of application-specific notions of edit distance. For example, by setting ϵ to 0 (*unit cost model*) users will get an edit script with the minimum number of edit operations, and by setting ϵ to 1 (*length cost model*) users will get an edit script with the minimum number of inserted or deleted edges.

2.3 PDiffView Prototype

PDiffview is available at http://www.seas.upenn.edu/∼zhuowei/diff/index.html. The prototype of PDiffview allows users to view, store and import/export SP-specifications and their associated runs. The user may then see the difference between two runs of the same specification by stepping through the set of edit operations in the minimum-cost edit script, or by seeing an overview (2.9). Since the graphs can be large, users may successively cluster modules in the specification to form a hierarchy of composite modules (following an extension of ZOOM). The difference between two runs of that specification can then be viewed at any level in the defined hierarchy, giving the user the ability to zoom in on composite modules that indicate a large amount of changes and ignore others that indicate no change (see Figure 2.10).



Specification: Protein annotation workflow
Source Run: R1 Target Run: R2
Cost Model: 1.0
Length : 4
Cost : 4.0
Current Operation 4/4
Type insert Cost 4
Path : extractDomSeq->getBrendaAnn

Property	Value
Name	R1
Vertices	19
Edges	23
Forks	
Loops	
Type	Node
Name	getDomA...
id	v2_8

Property	Value
Name	R2
Vertices	21
Edges	26
Forks	
Loops	
Type	Node
Name	getDomA...
id	v2_8

FIGURE 2.9 : Viewing the difference between R_1 and R_2 using PDiffView

C Related work

1 On the user view approach

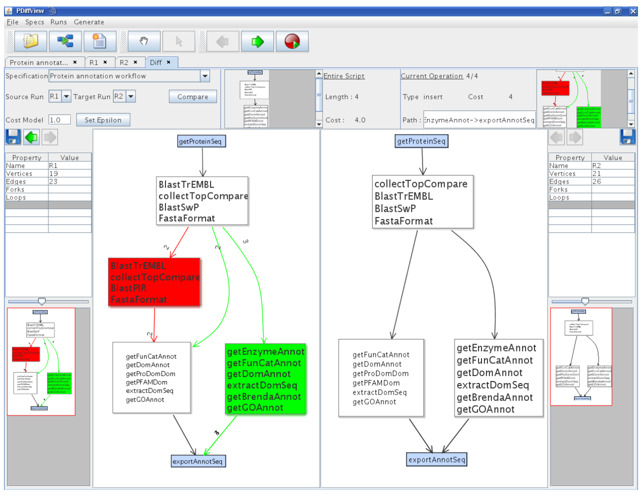
The ability to create composite tasks in scientific workflows (based on the ideas of Statecharts [Har87]) appears in many workflow systems: e.g., Kepler [BL05] allows composite actors, and Taverna/myGrid [ZWG+04] uses nested workflow processors/knowledge view models. Several approaches have considered the problem of querying (and some have also considered indexing) provenance information in the context of scientific workflow systems [OAF+03, BL05, ABL10, ABML09, MPB10]. However, none of them addresses the problem of querying through user views. The notion of *workflow summaries* [ABGK13] and *workflow abstractions* [ABL09] aim at reducing the complexity of workflows, which is also the aim of user views.

Constructing user views addresses a major problem and our work has a large impact. ZOOM has been adapted to hide private data [CCL+08, DKR+11]. Our model of provenance and user views is used in systems (e.g., PLUS Synthesizing Privacy, Lineage, Uncertainty and Security [BSM+08]). Other works consider the problem of retro engineering workflows [SLDC09] by redesigning composite modules to preserve provenance in the induced data flow. On a more formal point-of-view, a polynomial-time algorithm able to provide a minimum solution for ZOOM has been designed [BDKR09] (when considering a restriction in the graph structure of the workflow).

2 On the differing provenance approach

Within the context of scientific workflows, VisTrails [FSC+06] is one of the major approaches for differing workflows. However, VisTrails addresses the problem of differing workflows at the specification level while we deal with executions. Several other approaches focus on specifications including [KGH+12, SVK+07] which consists in *recording* the edit history between workflow versions, as well as computing deltas of RDF graphs [PTK+14, ZTC07](based on mechanisms proper to RDF), or [MvdAW08, vdAMW06] which develops a notion of process mining to compare different specifications.

PDiffView was the first approach to consider differing workflow runs. It is still the only polynomial-


PDiffView
File Specs Runs Generate
Protein annotat... R1 R2 Diff
Compare
Set Epsilon
getProteinSeq
BlastTrEMBL collectTopCompare BlastSwP FastaFormat
BlastTrEMBL collectTopCompare BlastPIR FastaFormat
getFunCatAnnot getDomAnnot getProDomDom getPFAMDom extractDomSeq getGOAnnot
getEnzymeAnnot getFunCatAnnot getDomAnnot extractDomSeq getBrendaAnnot getGOAnnot
exportAnnotSeq
collectTopCompare BlastTrEMBL BlastSwP FastaFormat

Specification Protein annotation workflow

Source Run R1 Target Run R2

Cost Model 1.0

Entire Script

Length : 4

Cost : 4.0

Current Operation 4/4

Type insert Cost 4

Path : EnzymeAnnot->exportAnnotSeq

Property	Value
Name	R1
Vertices	19
Edges	23
Forks	
Loops	

Property	Value
Name	R2
Vertices	21
Edges	26
Forks	
Loops	

FIGURE 2.10 : Viewing the difference at a high-level view using PDiffView

time algorithm able to compute the difference between two workflow executions (of the same SPFL workflow specification). PDiff [MWHW13] is a recent approach which differs runs represented by bipartite graphs (with data nodes and task nodes) and executed in the e-science central system.

More generally, related work includes differencing strings [AG97, Lev66, SK83, WF74], trees (ordered and unordered) [CRGMW96],[Tai79, ZS89, Kle98, Che01], and programs executions [Wil94, RBDL97, ZG05]). Our constrained edit operations preserve the validity of a run with respect to a given workflow specification which is not taken into consideration by any of the tree difference approaches.

As for the programs, usually static comparison of two program versions work at the binary level and cannot be used for workflow differencing.

D Supervision of students and collaborations

Modeling provenance in scientific workflow and designing ZOOM*userview was at the center of my postdoc work. I came with the idea of automatically designing composite modules by interviewing several workflow users, in particular phylogenetics from the pPOD project and oncologists from the HKIS European project (a project I participated to during my PhD). In both cases, the problem encountered by workflow users was to chain tools together, by adding (lots of) formatting steps, making the main relevant steps of the workflow difficult to get. I designed and implemented the ZOOM*userviews algorithm in collaboration with Susan Davidson and Olivier Biton. I then participated in the first provenance challenge to demonstrate the ability of our approach to deal with concrete scenarios. Later, while the first implementation of ZOOM was agnostic to the workflow system (expecting graphs representation of the specification), Heloise Bourlet (student from the Master of Bionformatics at Paris Sud I supervised during a 3 months internship) implemented a new module of ZOOM able to take in Taverna workflows.

A few months after the first provenance challenge, Zhuowei Bao[2] just started his PhD (supervised by Susan Davidson and Sanjeev Khanna). PDiffView was the first piece of his thesis work. I have actively worked with him on specifying the algorithm and carefully defining the structures of workflows on which

[2]Zhuowei Bao is now research scientist at FaceBook, Inc.

to focus, inspecting various examples. I have continued working on the differing workflow problematic with UPenn when I was back to France. Besides, I supervised Pierrick Girard (a 4th year master student from Polytech Paris-Sud) who spent a 4 month internship at UPenn in 2009 to design and implement the GUI of PDiffView.

E Conclusion and Perspectives

This Chapter presents my contributions to managing and querying provenance in scientific workflows.

The abstraction mechanism called *user views* for querying provenance information in workflows is introduced [BCBDH08]. The technique is appropriate for any data-oriented workflow system in which log-like information is collected at run time. The ZOOM algorithm constructs a user view based on input from the user on what modules they believe to be relevant. ZOOM guarantees that the view has one composite class for each relevant class, preserves and is complete w.r.t. the dataflow between relevant modules, and is minimal. We have designed a prototype for ZOOM [BCBD07] which have been used to participate to the first provenance challenge [CBBCD08].

Constructing user views addresses the major problem of dealing with provenance overflow. ZOOM is part of the first pools of papers published in the area of provenance in scientific workflows and has a large impact in the community. The main paper has 100+ citations while the demo paper has 33+ additional citations and the use of ZOOM in the context of the provenance challenge 45+. The main concepts are reused in various domains: secure views of in scientific workflows, (re)designing composite modules, modeling provenance in systems...

The second main piece of work of this Chapter addresses the problem of differing workflow runs of the same specification. Specifications are described by series-parallel graphs, overlaid with well-nested forks and loops. Forks and loops are captured as a laminar family of series subgraphs of the specification (loops may also be complete parallel subgraphs). After formalizing the notion of a valid run of a specification, the edit distance between a pair of valid runs is naturally defined as a minimum cost set of elementary path insertions and deletions that transform the first into the second run, and preserves the validity of each intermediate run [BCBD$^+$09]. The cost function used for each edit operation is compact yet general, allowing us to capture a variety of application-specific notions of distance, and depends on the start and end nodes as well as the length of the path. For this class of specifications, we present an efficient, polynomial-time differencing algorithm that relies on an equivalent formulation on SP-trees that are annotated with the fork and loop behavior. We have developed a prototype for PDiffView able to visualize the differences at various levels of granularity [BCBDG09].

Today's challenges in managing and querying provenance in scientific workflows are still numerous.

One of the most important challenges for end-users is to access workflow systems equipped with provenance modules able to deal with large-scale experiments (such as Next Generation Sequencing data). Until 2009, most of the workflow systems freely available to the scientific community were advanced research prototypes rather than systems able to work in production. Since 2010, Galaxy [GNT10] provides an open web-based system, particularly easy-to-use and robust. Such a system was at first quite basic while new functionalities are being constantly added to the system [Cle14] (e.g., subworkflows, running a workflow in a cloud) by the numerous contributors of the project. In the meantime, other kinds of very promising approaches have emerged and provide concrete solutions to make reproducible experiments (e.g., the ReproZip project [CSF13]).

In these projects, for the first time, very large amounts of provenance data start to be publicly available (although only a few provenance data sets as in the ProvBench initiative and the workflow specifications were previously shared). This places the problematic of provenance in scientific workflow within the big data paradigm [CCPR13, AFFW13]. Major challenges include indexing and managing large-scale distributed provenance information [dOOBM12, VRJ+13, DZG+12], accessing and analysing big provenance data sets [TKSP12] while representing uniformly data and workflow provenance from distributed and heterogenous environments [ADD+11, DKBL12]. This also opens the door to new research questions where provenance information is exploited to query more accurately workflows repositories by using the similarity between data sets, or to tune more efficiently schedulers in cloud systems by designing adaptive scheduling based on previous executions [BL13, COM14]...

Sommaire

CHAPTER

3

WORKFLOW SIMILARITY AND REWRITING APPROACHES TO ENHANCE WORKFLOW REUSE

Main Publications of the Chapter

[SCBL12]	**(Re)Use in Public Scientific Workflow Repositories**, SSDBM 2012
[SBCBL14]	**Similarity Search for Scientific Workflows**, VLDB 2014
[SCBK+14]	**Layer Decomposition: An Effective Structure-Based Approach for Scientific Workflow Similarity**, eScience 2014
[CCBF+14]	**DistillFlow: removing redundancy in scientific workflows**, SSDBM 2014 (Demonstration)
[CBCG+14]	**Distilling structure in Taverna scientific workflows: a refactoring approach**, BMC Bioinformatics 2014.
[CBFC12b]	**Scientific workflow rewriting while preserving provenance**, eScience 2012

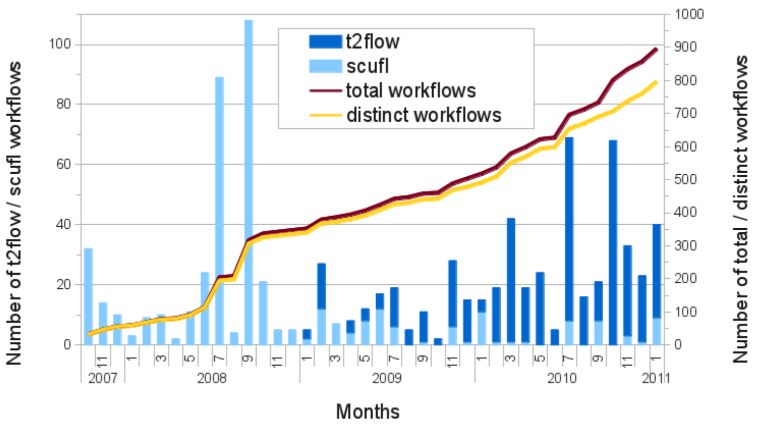


FIGURE 3.1 : Taverna workflows uploaded to myExperiment by month (left hand scale), and the numbers of overall and distinct workflows in the myExperiment repository the uploads amount to (right hand scale). We consider the two Taverna formats *scufl* and *t2flow*.

One of the major promises for users when using workflows to design their experiments is that workflows are easier to share and reuse than scripts. However, on concrete situations, workflows do not appear to be reused and shared as much as they could be. A large part of my work in the last years has been dedicated to the topic of workflow reuse and more precisely to understand (i) how (and how much) do authors share their workflows, (ii) why users may not be able to reuse workflows as much (and as frequently) as they wanted to and (iii) what can be done to enhance scientific workflow reuse. The originality of my approaches lies in interviewing users to fit their needs while considering graph techniques on workflows to address the depicted problems.

More specifically, the outline of this Chapter is the following. The first section presents the results of our study on scientific workflow reuse based on the public information available on myExperiment (Section A). Solutions developed to two complementary problems related to the low level of reuse are then introduced, namely, (i) providing better means to query scientific workflow repositories and (ii) reducing the complexity of workflow structures. For the first point, we introduce a large comparative framework for workflow similarity and one efficient structure-based solution for workflow retrieval. As for point (ii), Section C introduced our two approaches reducing the structural complexity of workflows, namely, DistillFlow and SPFlow.

A Study of workflow reuse

1 Motivations

We study the reuse of workflow elements in myExperiment.org, the largest and most accessed public scientific workflow repository, which contain in majority Taverna bioinformatics workflows. Figure 3.1 shows the number of Taverna workflows submitted to myExperiment per month and the total number of workflows available.

Objects of study. A Taverna workflow is defined by one top-level dataflow and may contain nested dataflows (while the workflow is defined by the dataflows plus some metadata associated to it including

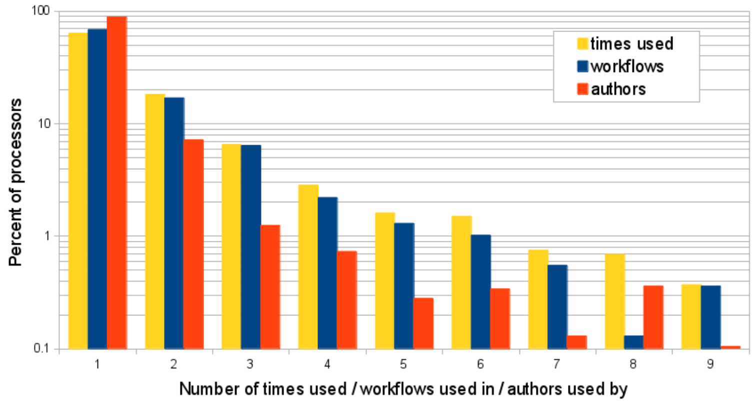


FIGURE 3.2 : Usage of processors overall, in workflows and by authors.

title, description, and author). Each dataflow has its own main inputs and outputs and consists of one or more processors representing activities operating on data. Processors are connected by datalinks, describing the flow of data from one processors output(s) to the next processors input(s). Each processor 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.

The first step of our study is to determine methods to identify workflow elements at each of the three levels, i.e., at processor level, at dataflow level, and at workflow level.

For processors, we use the configuration file (containing the script of the processor or the web service call) to identify them. Both myExperiment and Taverna provide identifiers for dataflows which can be overlapping. We choose to identify dataflows by the (multi)sets of processors they contained. We follow the same process for comparing workflows: two workflows are identical if they are built from the exact same set of dataflows and processors.

2 Results of our study

On average, each workflow uses 1.6 dataflows and 11.4 processors, where the largest workflow has 455 processors and the smallest has 1 processor. In terms of dataflows, the largest workflow contains 19 nested dataflows. We also find 3,598 of 10,242 processors, 1,071 of 1,431 dataflows and 792 of 898 workflows to be distinct.

2.1 Reusing Processors

We first look at the general usage of processors by comparing the total numbers of processor occurrences with the number of distinct processors, and their use across workflows and authors. On average, each processor is used 2.85 times in 2.24 workflows by 1.31 authors. Figure 3.2 shows the relative usage frequencies for all processors in our set. Overall reuse of processors and cross-workflow reuse of processors closely correlate. 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.

We more precisely consider usage statistics with the processor's category and subcategory, which allows to underline three points.

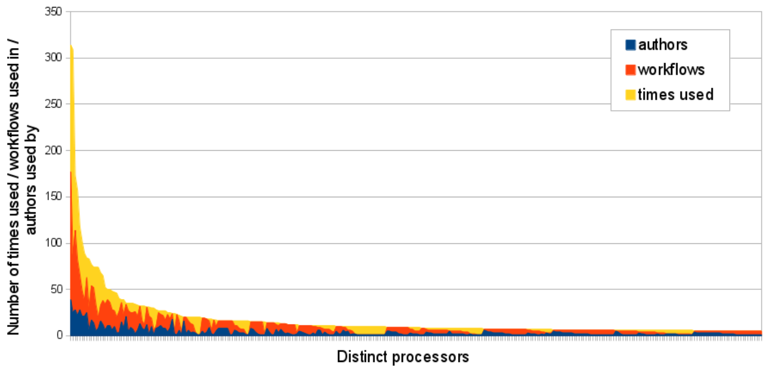


FIGURE 3.3 : Usage counts of the 300 most used processors showing a Zipf-like distribution.

- Processors from the ***local*** category (belonging to the library of tools provided by Taverna) typically perform format changes, or technical operations, they allow to glue processors together in a workflow. 27% of such processors are distinct with a high reuse rate of 44%.

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

- ***Web-service*** processors show 39% distinctiveness, and 44% reuse. By far the most popular types of web-service invocations are *soaplab* and *wsdl* processors. 24% of *soaploab* processors and 39% of *wsdl* processors 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 SimpleSearch*.

Figure 3.3 shows the 300 most frequently used processors and their cross-workflow and cross-author reuse. Overall usage counts clearly follow a Zipf-like distribution. Zipf's law [Zip35] 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 processors in scientific workflows, it means that only few processors are used very often, while usage of the vast majority of processors is very sparse.

2.2 Reusing Workflows and Dataflows

A last series of experiments showed that over 80% of dataflows are used only once, and only 5% used more than twice. 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.

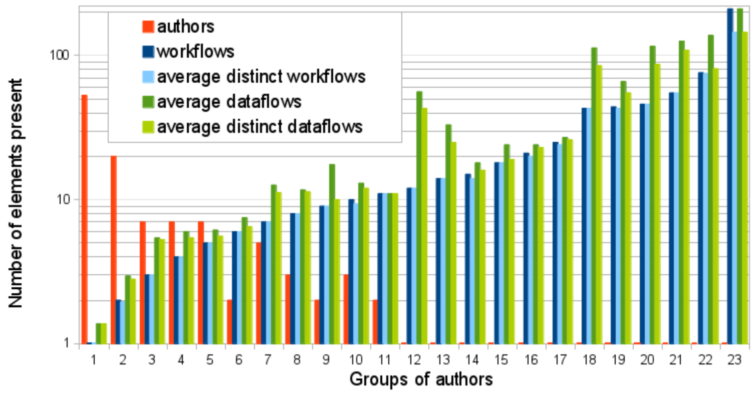


FIGURE 3.4 : Authors grouped by the total number of workflows they have created. Total amounts of workflows, and averages of distinct workflows, and total and distinct dataflows shown for each group.

Of the 898 workflows, 83 appear more than once in the repository, 19 of which were 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 3.4 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 this user duplicates are caused by equivalent workflows being uploaded in both *scufl* and *t2flow* formats. Figure 3.4 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 processors 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.

Another important 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.

2.3 Main Conclusion

While reuse is already performed by some users, this study shows that cross-author reuse is low and could be increased. Following the study we performed interviews of several users of workflows repositories which reveals that (i) there is a crucial need for adequate means to retrieve existing workflows from repositories, and (ii) some workflows have too complex structures to be reused.

The following two sections provide our contributions to these two points.

B Workflow similarity approaches

1 Motivation

Designing a query language on top of a workflow repository or classifying the large amount of available workflows in a repository are challenging points to enhance reuse. They are directly related to the problem of similarity between workflows.

Our contributions to the topic of workflow similarity are twofold. First, we design a comparison framework to precisely evaluate the ability of workflow similarity approaches to retrieve similar workflows in large repositories. Second, we introduce a new structure-based while effective workflow similarity approach.

As an illustration, Figure 3.5 provides an example of a pair of workflows. While the number and name of their inputs and outputs are not the same and the size and names of the workflow processors are different, such workflows may perform quite a similar task. A closer look shows that both workflows actually provide information on pathways and they share some processors (get_pathways_by_genes, get_image_from_URL). It may also be the case that module names are different while the processors are actually the same (that is, they have the same source code). Extra annotation on the workflow (tags or free text descriptions) may also be considered to determine similarity. Determining whether or not two workflows are different may thus involve a large variety of approaches, using various kinds of information.

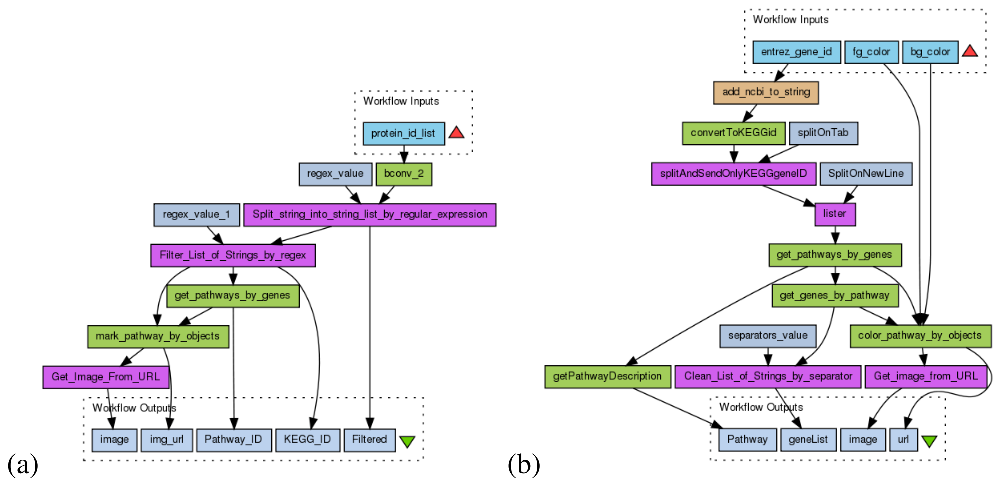


FIGURE 3.5 : Sample scientific workflows from the myExperiment repository: (a) ID: 1189, Title:KEGG pathway analysis, (b) ID: 2805, Title: Get Pathway-Genes by Entrez gene id.

2 Review of workflow similarity approaches

2.1 A framework for workflow similarity

Our first contribution is the design of a framework made of elementary bricks in which any existing approach for workflow similarity could be described, yet, opening the door to new (hybrid) solutions. Figure 3.6 introduces such a framework.

Based on a large review of the literature, we distinguish two families of approaches for performing workflow similarity: annotation-based (bottom of the figure), using the meta data associated to the workflow and structure-based (upper part of the figure), using the graph structure of the workflow to perform the comparison. We describe below briefly the various approaches considered.

In the first family (annotation-based), information exploited includes the workflow's title, a free form text description, and assigned keyword tags. Only two kinds of such approaches can be considered, described here after.

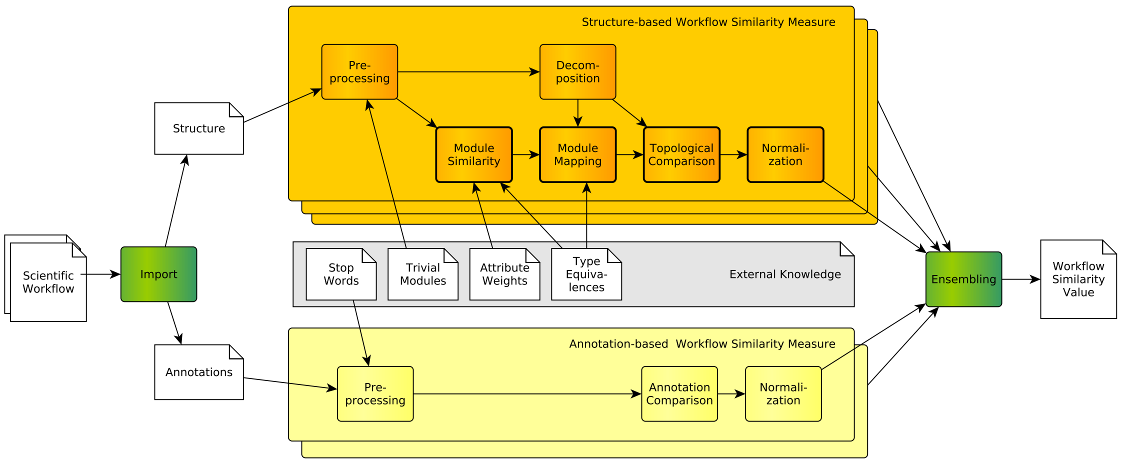


FIGURE 3.6 : Scientific workflow similarity framework

1. **Bag of Words** – In this approach, following the work of [COO+10], workflows are compared by their titles and descriptions using a bag-of-words techniques. Both title and description are tokenized using whitespace and underscores as separators.

2. **Bag of Tags** – The keyword tags assigned to scientific workflows in a repository can also be used for similarity assessment, as done in [STD10]. 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.

The second family of approaches (structure-based) is obviously more complex and we decompose the process of structural comparison into fours steps: module identification, module mapping, topological comparison and normalization. For each, we list the various techniques used by approaches found in the literature. Table 3.1 provides all the compared configurations (see [SBCBL14] for more information).

Topological comparison may be the most computationally expensive step. Approaches can be classified as either a) structure agnostic, i.e., based only on the sets of modules present in two workflows, [SCM+11, SLA+08, STD10, BG12]; b) based on substructures of workflows, such as maximum common subgraphs [SLA+08, GLG06, FR10] or graph kernels derived from frequent subgraphs [FR10]; or c) using the full structure of the compared workflows [XM07].

In our study, we include an approach to topological comparison for each of these classes, described here-after.

- **Sets of Modules** – Analogous to the similarity measure described in [SCM+11, SLA+08, STD10, BG12], two workflows $wf1$ and $wf2$ are treated as sets of modules.

- **Sets of Paths** – As a slightly relaxed version of using the maximum isomorphic subgraph for workflow comparison [SLA+08, GLG06, FR10], the sets of all paths two DAGs are comprised of can be used to compare them by their maximum similar subgraph [Kri01]. 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.

- **Graph Edit Distance** – We call Graph Edit Distance an approach analogous to the work presented in [XM07], where the full DAG structures of two workflows are compared by computing the graph edit distance using the SUBDUE [MB00] 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 SUBDUE's input format. For computing graph edit distance, we kept SUBDUE's default configuration which defines equal costs of 1 for any of the possible edit operations. Testing several different weighting schemes did not produce significantly different results.

2.2 Constituting a Gold Standard

To compare the various approaches we have constituted a gold standard. We have conducted a user study to collect manually assigned similarity ratings for selected pairs of scientific workflows. Overall, 15 domain experts from seven different institutions participated including members of the University of California Davis, the University of California Los Angeles, the University of Barcelona, the University of Manchester, the University of Humboldt zu Berlin (Germany), the University of Paris Sud (Institut de Genetique et Microbologie, France), the Institut pour la Recherche et le developpement (IRD, France).

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 w.r.t. a given query workflow using a naive annotation-based similarity measure and drew workflows at random from the top-10, the middle, and the lower 30. Life science workflows of this set mostly deal with genomic and proteomic sequences similarity, functional annotation of pathways, microarrays normalization and analysis processes.

The ratings were to be given along a four step Likert scale [Lik32] with the options *very similar*, *similar*, *related*, and *dissimilar* plus an additional option *unsure*. *Unsure* user ratings were not further considered in the evaluation. We asked users to perform their rating based on the functionality of the workflows: two workflows are similar iff they can be used to perform the same kind of analysis or to answer to same kind of biological question. 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 [CBDH11] (which will be described in the next chapter), extended to allow incomplete rankings with *unsure* ratings. On the basis of the generated consensus rankings, we evaluate the algorithms' *ranking correctness*. Figure 3.7 inspects inter-annotator agreement, comparing each single expert's rankings to the generated consensus using the ranking correctness and completeness measures described in Chapter 4. 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 1,483 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

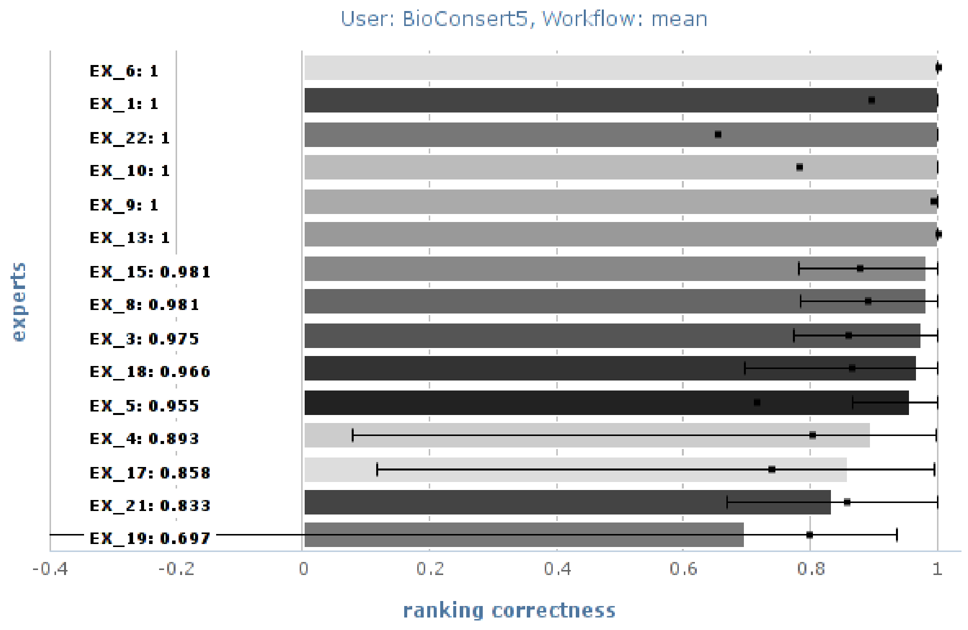


FIGURE 3.7 : Mean ranking correctness (bars) with upper and lower stddev (errorbars), and mean ranking completeness (black squares) for single experts' rankings compared to the ranking derived as BioConsert expert consensus.

in the search results of each of the used algorithms w.r.t. their user-perceived similarity. Using these completed ratings we evaluate the algorithms' *retrieval precision* used in Figure 3.8.

2.3 Overview of the results on similarity for workflows

We summarize below the main results obtained by our study. Please consider Figure 3.8 without looking at LD lines for now, and table 3.1 to get an overview of all the algorithms and configurations tested.

- As for techniques to identify and compare modules, 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 [SHI].

- 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 inspected here, or the CrowdLabs repository of VisTrails workflows [MSFS11], 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 expensive 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. This will be the topic of the next section.

- Normalization of the similarity values derived from workflow comparison w.r.t. workflow size is, as clearly shown, indispensable for similarity search of scientific workflows.

- 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. More precisely, we have considered the following two techniques.

  – *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. 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.

- Using ensembles of different algorithms (combos) can significantly improve result quality when compared to single algorithms.

3 Introducing the Layer Decomposition approach

To address the need for considering efficient structural approaches for workflow similarity, we introduce a novel approach, called Layer Decomposition (LD), for structurally comparing two workflows.

The fundamental idea behind Layer Decomposition 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 to reduce the cost of the comparison. For instance, it is completely lost when only module sets are compared, and a few graph edits can lead to workflows with very different

TABLE 3.1 : Algorithm shorthand notation overview

Notation	Description
MS	*Module Sets* topological comparison
PS	*Path Sets* topological comparison
LD	*Layer Decomposition* 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

execution orders (like swapping the first and last of a long sequence of modules). Second, we observed in our previous evaluation 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 impacts 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.

Back to the results provided by Figure 3.8, the ranking performance for simLD in direct comparison to simMS, simPS, simGE, and the annotation based measures simBW and simBT can be seen. Results have been sorted by mean ranking correctness.

Several points can be underlined.

- Firstly, simLD provides best results.

- Secondly, both simLD and simPS provide most stable results across different configurations. Performance of simMS, 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 (importance projection).

- Thirdly, while simMS, when configured properly, can achieve ranking correctness values comparable to the best results of simLD, simPS is generally slighly behind simLD. In contrast, simGE, 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 simLD and simPS fully distinguish all workflows in terms of their similarity to the query workflows where users make a distinction as well.

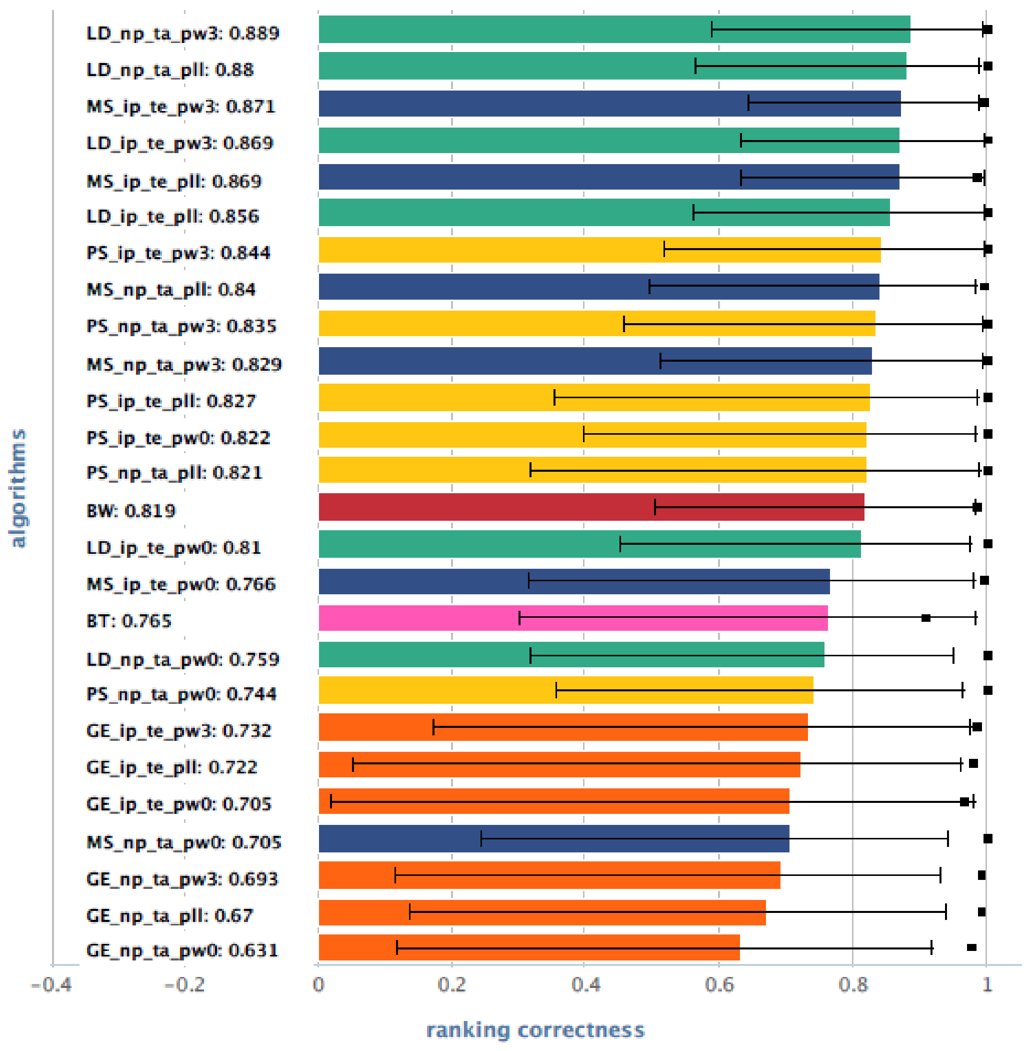


FIGURE 3.8 : Mean ranking correctness (bars) with upper and lower stddev (errorbars), and mean ranking completeness (black squares) over 24 lists of 10 workflows for different algorithms and configurations (see Table 3.1 for notation).

C Reducing the structural complexity of workflows

To enhance workflow reuse, we address a second kind of problem: reducing the structural complexity of workflows. The idea here has a dual objective: (i) reduce (even visually) the structural workflow complexity to make workflows looking simpler, easy to "grasp" by users, and (ii) make the complexity of (inherently complex) graph-based operations lower, because they are performed on simpler structures. While we made contributions related to the first point in DistillFlow, by removing redundancy in workflow to provide more compact structures, we made contributions associated to the second point in SPFlow by providing an algorithm able to transform any non series-parallel workflow graph into a series-parallel workflow structure.

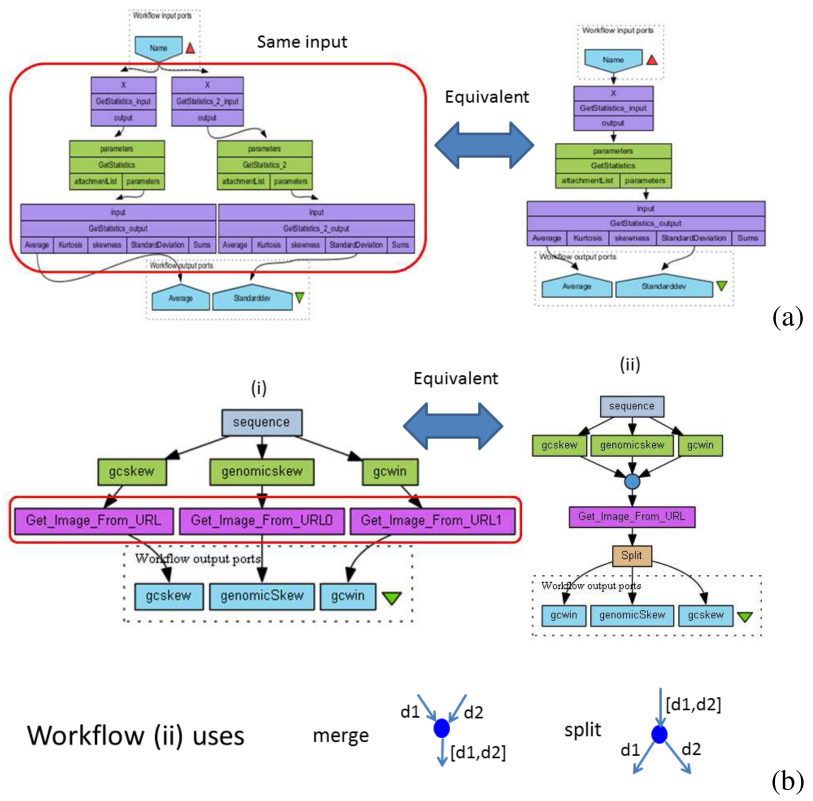


FIGURE 3.9 : Two use cases.

1 Distilling the structure of workflows with DistillFlow

1.1 Motivation and use cases

As usual, we start this work by carefully inspecting the complex structure of Taverna scientific workflows. Such an inspection reveals that, in numerous cases, the structure complexity is due mainly to redundancy. More specifically, 26 % of such workflows have explicit redundancy in their structure (e.g., several occurrences of the same processor with the same input data). Authors of such workflows include users not familiar with programming code reuse (copy pasting processors as they could have copy pasted parts of code) and users who are not aware of the powerful "implicit iteration process" of Taverna (that is, they are not aware of the techniques used by the Taverna system to deal with for-each loops). A completely different kind of user is also targeted such as users who duplicate tasks to optimize their workflows (while this kind of optimization should be performed at the execution time by the workflow engine and not by the user at the specification level). Notably, such a behavior is not particular to Taverna users and can be found in users of other workflow systems. However, as we propose to concretely rewrite workflows, we have to consider one workflow system and we have chosen Taverna.

In turn, redundancy may be an indicator of over-complicated design, and thus there is a chance for a reduction in complexity which does not alter the workflow semantics. Our main contention in this work is that such a reduction in complexity can be performed automatically, and that it will be beneficial both in terms of user experience (easier design and maintenance), and in terms of operational efficiency (easier to manage, and sometimes to exploit the latent parallelism amongst the tasks).

We first describe the two kinds of situations found, captured by two use cases.

The first use case (Figure 3.9 (a)) is represented by a workflow performing a very basic statistical analysis of a data set (based on integer values). This workflow involves the duplication of a linear chain of connected processors $GetStatistics\_input$, $GetStatistics$ and $GetStatistics\_output$. The last processor in the chain reveals the rationale for this design, namely to use *one output port from each copy of the processor*. Clearly, this is unnecessary, and the version in Figure 1 (ii) achieves the same effect much more economically, by drawing both output values from the same copy of the processor.

In the second use case (Figure 3.9 (b)), the representative workflow makes use of the G-language Genome Analysis Environment[3] to calculate three features of a given genome sequence. It begins with three distinct processing steps on the same input sequence. We observe that the three steps that follow those are really all copies of a master $Get\_image\_From\_URL$ task. This suggests that their three inputs can be collected into a list, and the three occurrences can be factored into a single occurrence which consumes the list. By virtue of the Taverna list processing feature, the single occurrence will be activated three times, one for each element in the input list. Also, the outputs of the repeated calls of $Get\_image\_From\_URL$ will be in the same order as items in the list. Therefore this new pattern achieves the same result as the original workflow. Note that collecting the three outputs into a list requires a new built-in *merge* node (the circle icon in Figure 2 (ii)). Similarly, a *Split* processor has been introduced to decompose the outputs (list of values) into three single outputs.

1.2 Anti-patterns and refactoring approach

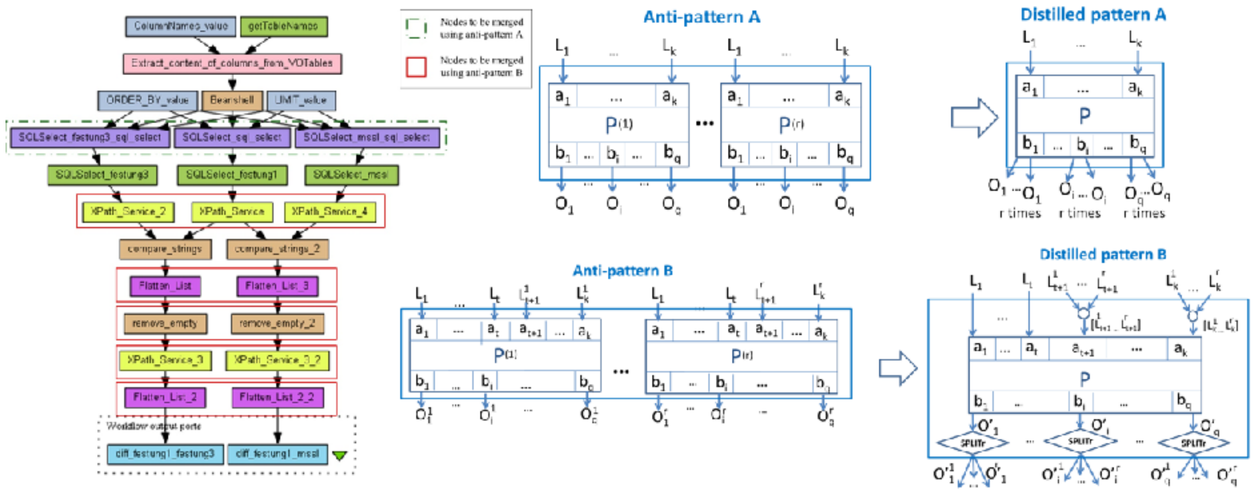

Nodes to be merged using anti-pattern A
Nodes to be merged using anti-pattern B
Anti-pattern A
Distilled pattern A
Anti-pattern B
Distilled pattern B
SPLIT
r times

FIGURE 3.10 : Workflow where anti-patterns have been highlighted; Rewriting rules for anti-patterns of kinds A and B.

In response to these findings, our approach provides a refactoring procedure to replace several occurrences of the same processor with one single occurrence whenever possible, while following several assumptions: (i) the processors we consider are deterministic (the same output is produced given the same input); (ii) only processors implemented using the exact same code can be merged (in our setting, two processors are equivalent if they represent identical web service calls, or they contain the same script, or they are bound to the same executable Java program). In practice, condition (ii) is often realized, because processors are duplicated during workflow design by means of a graphical "copy and

[3]http://www.g-language.org/wiki/

paste" operation. (iii) Only copies of processors that do not depend on each other's can be merged (no directed path can be found between each two copies).

Figure 3.10 introduces the two generic patterns and the transformation rules we have considered; these rules allow the rewriting of anti-patterns into patterns without redundancy. Note that each anti-pattern depicted actually covers a large number of concrete cases of redundancy (in particular those two generic patterns and rewriting rules are able to refactor all the workflows with explicit redundancy detected). More precisely, anti-pattern A deals with simple redundancy where the same processor P appears with r occurrences (denoted $P^{(1)}...P^{(r)}$ in the figure) and each occurrence takes the exact same input (and can be thus replaced by one single occurrence) and produces the exact same output. Anti-pattern B depicts a more complex case where the various occurrences of the same processor have only part of their inputs in common. In the rewritten pattern, input data that differ from one occurrence to another (L^l_{t+1} to L^l_k) have been merged using the merge processors provided by Taverna (the circle icon) to construct lists of data from the original data items to exploit the implicit iterative process of Taverna. As a consequence, the rewritten pattern contains a *list split* processor called $SPLIT_r$ to decompose the list obtained as output into r pieces to ensure that the downstream fragment of the workflow remains unchanged. When detecting the anti-patterns, we looked into the properties of each processor, including processor types, inputs, outputs, web services calls, script implementation etc., to make sure that the processors are exactly the same.

The DistillFlow algorithm removes as many anti-patterns as possible, following a recursive process. Figure 3.11 (a1) provides the representation of the initial workflow depicted in Figure 3.10 while (b1) shows the distilled workflow. We omit here the details about the way we carefully apply rewriting rules and how we consider the various strategies followed by the processors when dealing with lists of input data (cross and dot strategies, see [CBFC12a] for more information on this point).

Evaluation. To evaluate our approach, we consider two sets of workflows: the myExperiment data set (which contained 1,400 workflows in 2013) and a set of 70 private, very carefully designed Taverna workflows, from the BioVel project. BioVel (www.biovel.eu) is a consortium of fifteen European partners from nine countries which aims at developing a virtual e-laboratory to facilitate research on biodiversity. BioVel promotes workflow sharing by using the Taverna system and aims at providing a library of workflows in the domain of biodiversity data analysis. Access to the repository to contributors, however, is restricted and controlled. Because of the restricted access and the focus on a specific domain of these workflows, they are expected to be of higher quality than the general myExperiment workflow population.

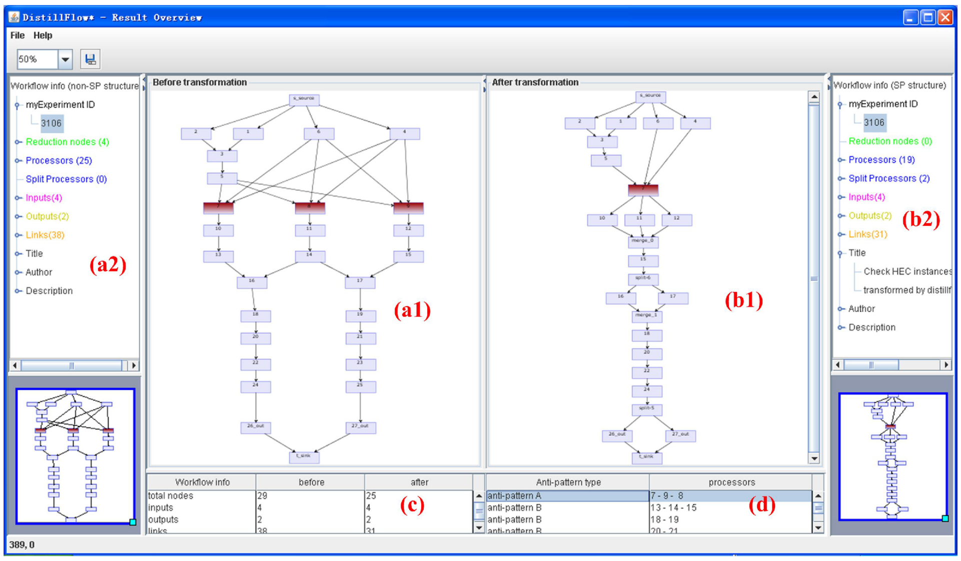
As a result, 25.7% of the workflows of the myExperiment set contains at least one anti-pattern. Although anti-pattern A appears in only 5.5% of the total, it is particularly costly because it involves multiple executions of the same processor with the exact same input, therefore being able to remove it would be particularly beneficial.

As for the BioVel private workflows, 40.8% include at least one anti-pattern, while all anti-patterns are of kind B (and thus none contains any kind A). Additionally, we observe that on average a workflow from BioVel contains fewer anti-patterns than a workflow from myExperiment.

1.3 The DistillFlow Prototype

DistillFlow has been implemented in Java. It includes both a library of anti-patterns and a library of rewriting rules (algorithms) to consider more anti-patterns and rewriting rules in the future. Users communicate with DistillFlow by loading and interacting with original and rewritten workflows.

Figure 3.11 provides a snapshot of the DistillFlow user interface. Users start using DistillFlow by loading a workflow specification into the system. The anti-patterns are detected and can be removed either all-at-once or in a step-by-step process (selected by the user). Both graphs (initial and transformed) can be visualized.



Workflow info (non-SP structure)
- myExperiment ID
  - 3106
- Reduction nodes (4)
- Processors (25)
- Split Processors (0)
- Inputs(4)
- Outputs(2)
- Links(38)
- Title
- Author
- Description

Workflow info (SP structure)
- myExperiment ID
  - 3106
- Reduction nodes (0)
- Processors (19)
- Split Processors (2)
- Inputs(4)
- Outputs(2)
- Links(31)
- Title
  - Check HEC instances
  - transformed by distillf
- Author
- Description

Workflow info	before	after
total nodes	29	25
inputs	4	4
outputs	2	2
links	38	31

Anti-pattern type	processors
anti-pattern A	7 - 9 - 8
anti-pattern B	13 - 14 - 15
anti-pattern B	18 - 19
anti-pattern B	20 - 21

FIGURE 3.11 : Visualizing both initial workflow and distilled workflow.

2 Reducing the inherent complexity of workflows with SPFlow

We now consider another kind of workflow complexity which is not related to the number of edges and nodes but rather to the intricate nature of the workflow structure. More formally, a workflow is considered as complex if useful graph operations, in particular involving sub-graph isomorphism, cannot be performed by a polynomial algorithm. We consider here a rewriting approach able to transform any workflow having a DAG structure into a series-parallel workflow. Our approach ensures that the provenance of any data produced by a rewritten workflow is the same as the provenance it would have had in the original workflow. In other words, we consider a *provenance-equivalent* rewriting approach. The main motivation for such an approach is the increasing presence of solutions dedicated to series-parallel graphs to deal with provenance information while workflows added to repositories have increasingly complex structures.

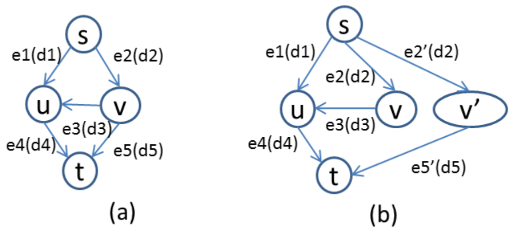


FIGURE 3.12 : Two graphs illustrating provenance related notions

We consider again the notion of series-parallel (SP) graphs. Determining whether a graph has an SP structure can be performed in linear time [VTL79]; this process is based on a set of reduction operations introduced in [Val78, BKS92], and illustrated in Figures 3.13 and 3.14. Duffin proved that a given DAG is series-parallel if and only if it does not contain a subgraph homeomorphic to the "forbidden subgraph" of Figure 3.12 (a).

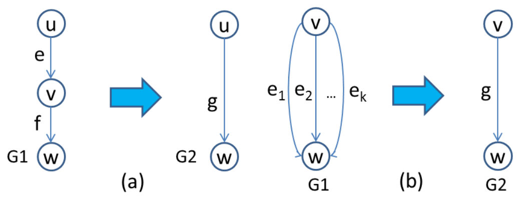


FIGURE 3.13 : (a) Series reduction; (b) Parallel reduction

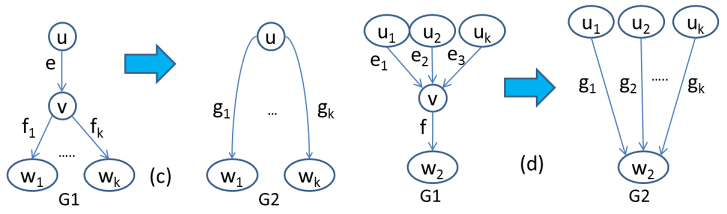


FIGURE 3.14 : (a) Out-vertex reduction; (b) In-vertex reduction

A graph is said to be **maximally reduced** if and only if no more series or parallel reduction can be applied to it. A graph is **SP** if and only if there exists a sequence of series and parallel reductions that reduces G to BSP (i.e., no node reduction). Figure 3.15 describes reductions performed on a graph. No vertex reduction is used to get the BSP graph: the initial graph G_0 is SP.

The number of times vertex reduction operations are used can give an idea of the *distance* between a non-SP to an SP structure. Intuitively, the highest number of times vertex reductions are used the farthest from an SP structure it is.

2.1 Provenance-equivalence

The provenance of a piece of data labeling an edge is the ordered sequence of tasks performed to produce this data, and input data to each task. Immediate provenance describes the last step of production while deep provenance describes the entire sequence of steps that produced the data [BCBDH08].

In the present work, we represent provenance by using expressions obtained from the labels of edges in the graphs which are reduced following the process described above. Such expressions allow to keep track of the subgraphs they replace. More precisely, labels are actually expressions built on the nodes and edges labels, and using two associative binary operators "+" (when parallel reductions are used) and "·" (when series reductions are used).

The **Immediate Provenance** of a final output f in a workflow run is defined by the label of the last processor that has been run (e.g., $\tilde{u}$) and the labels of the input data such processor took in (e.g., $d_1 \dots d_p$). $imProv(f) = \tilde{u} \cdot (d_1 + \dots + d_p)$. In the same context, the **Deep Provenance** of the final output f of a run is recursively defined by $DProv$ with $DProv(f) = \tilde{u} \cdot (d_1 \cdot DProv(e_1) + \dots + d_p \cdot DProv(e_p))$

The base case occurs when u is the source of the graph and f is an outgoing edge of the source: $DProv(f) = imProv(f) = \tilde{s}$.

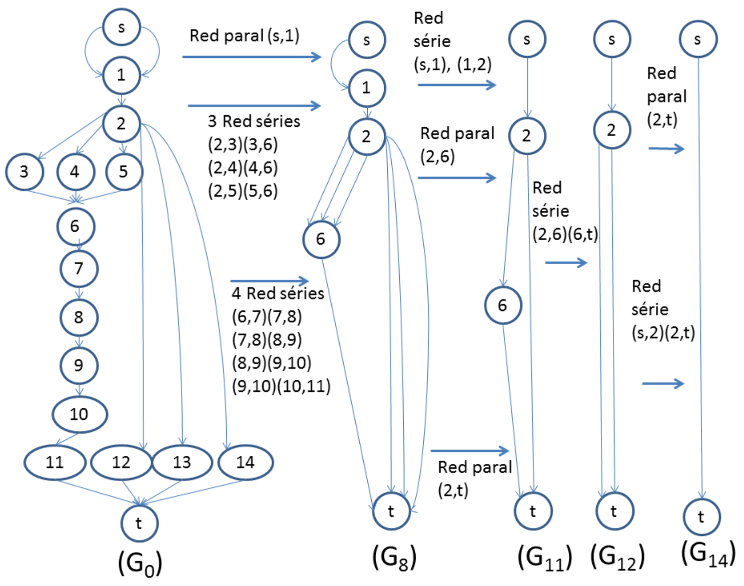


FIGURE 3.15 : Example of reduction operations applied to G_0.

There is several equivalent expressions for provenance due to associativity, commutativity and distributivity properties. Indeed, "+" is commutative because several parallel input data can be considered in any order. However, "·" is not commutative since the order in which tasks are executed is important. Right distributivity can be used to provide a concise representation of provenance through the following **factorization rule**: $(\alpha_1 \cdot z \cdot \beta + \alpha_2 \cdot z \cdot \beta) \rightarrow (\alpha_1 + \alpha_2) \cdot z \cdot \beta$ where α_1, α_2 and β are expressions built on vertex and edge labels using "+" and "·" and z is a vertex label.

Note that given a run G_{run} and a vertex u, all the outgoing edges of u have the same provenance, as they are all outputs of the same task.

Example 3.1 *Consider the graph G_r of Figure 3.12 (a). The immediate provenance of data d_5 flowing in edge e_5 is $imProv(e_5) = \tilde{v} \cdot (d_2 + d_3)$. We also have $DProv(e_5) = \tilde{v} \cdot (d_2 \cdot DProv(e_2) + d_3 \cdot DProv(e_3)) = \tilde{v} \cdot (d_2 \cdot \tilde{s} + d_3 \cdot [\tilde{u} \cdot d_1 \cdot \tilde{s}]) = \tilde{v} \cdot [d_2 + (d_3 \cdot \tilde{u} \cdot d_1)] \cdot \tilde{s}$.*

Our aim is to transform a run into another which has an SP structure while preserving provenance. For this, we require the two graphs to have the same provenance for their latest output (actually the union of all outputs when several outputs are involved, see [CBFC12b] for more details). We thus introduce here the notion of **provenance-equivalence** of two graphs (or in our setting, of two workflow runs) and state that the reduction operations rewrite the initial graph into a provenance-equivalent graph.

Definition 3.1 *Let G_{r1}, G_{r2} be two runs. G_{r1} and G_{r2} are **provenance-equivalent**, noted $G_{r1} \overset{prov}{\Leftrightarrow} G_{r2}$, if and only if the deep provenance of their final outputs is the same.*

Graphs (a) and (b) of Figure 3.12 are provenance-equivalent.

2.2 Major features of the SPFlow Algorithm

Based on these definitions we can show that the reduction operations preserve provenance. Additionally, among the strategies found in the literature to rewrite non-SP graphs into SP graphs (while never applied to scientific workflows) none are able to preserve provenance, that is, none were able to provide provenance-equivalent graphs. We thus introduce our own strategy, based on in-vertex duplication as depicted in Figure 3.12, and can show that our approach is provenance-preserving.

As a consequence, the SPFlow algorithm rewrites a non-SP graph G into a new SP graph, called SPG obtained from G by duplicating vertices of G, while ensuring that G and SPG are provenance-equivalent. As expected, SPFlow has an exponential complexity in the worst case (which occurs in a graph consisting of n iterations of the forbidden subgraph depicted in Figure 3.12). To evaluate the benefit of using SPFlow in real settings, we again consider the set of Taverna workflows available in myExperiment. Although three graphs have an important number of duplicated vertices, the very large majority of graphs, including huge workflows (having more than 100 vertices), have a small ratio, lower than 5. Additionally, the time to rewrite each workflow is negligible for the current structures of workflows: on a dual core@2.2GHz and 2GB of RAM desktop, the maximum time is 434 ms.

D Related work

1 Workflow reuse

Only a very few studies have considered 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 [TZF10], who find that only few web-services are used in more than one workflow. Yet, another analysis of processors contained in the workflows stored in myExperiment [WVDVW$^+$09], reveals that the majority of basic tasks are local processors and not web services. In [STD10], functional properties of several types of processors, including local ones, are used as features to classify workflows. Yet, no differentiation is made between single types of processors after extraction, hindering fine-grained analysis of shared elements.

Our study of workflow reuse is the first to provide a comprehensive account on the degree of sharing between workflows in a repository. In particular, three key aspects are taken into account: (1) considering alternative methods to test the identity of (or similarity between) two elements from different workflows; (2) describing workflows at three levels of functionality: single processors (typically treated as black boxes), groups of processors organized as dataflows, and whole workflows; (3) considering authorship.

2 Scientific workflow similarity

As for the review of the methods for scientific workflow comparison, a few studies are available and the concrete type of evaluation the proposed methods are subjected to varies with the intended use case, e.g., clustering or similarity search. Additionally, evaluation settings vary greatly from manual inspection of a method's output [COO$^+$10, STD10, SCM$^+$11, GLG06] to systematic evaluation against a small (and not publicly available) set of expert provided truth [BG12] or against ground truth derived from another method [FR10]. Drawing conclusions on these studies is not possible. We provide a unique framework able to capture all the techniques available, combine them and compare them using two large, public and independent data sets (workflows from myexperiment and the Galaxy repository) while considering a gold standard (made public) based on the rating of 15 experts.

As for the rewriting approaches, we are the first to consider the notion of redundancy in workflow and consider a refactoring approach based on the study of workflow structure. More research is available from the business workflows community, where several analysis techniques propose to discover control-flow errors in workflows (see [vdAvHtH+11] for references). Other work in this community focus on data-flow verification [TAS09]. However, such work aimed primarily at detecting access concurrency problems in workflows using temporal logics, making both aims and approach different from ours. Also, it would be hard to transfer those results to the realm of scientific workflows, which are missing the complex control constructs of business workflows, and instead follow a dataflow model (a recent study [MGRtH11] has shown that scientific workflows involve dataflow patterns that cannot be met in business workflows).

With the increase in popularity of workflow-based science, and bioinformatics in particular, the study of scientific workflow structures is becoming a timely research topic. Classification models are developed to detect additional patterns in structure, usage and data [RP10, YGN09b]. More high-level patterns, associated to specific cases of use (data curation, analysis) are identified in Taverna and Wings workflows in [GAB+12]. Complementary to this work, graph-based approaches are considered for automatically combining several analysis steps to help the workflow design process [RMMTS12].

As for the last piece of work described in this Chapter (SPFlow), we first review the graph literature and consider the major NonSP to SP rewriting approaches, namely (Re)synchronization [GE03] and in-vertex duplication [BKS92]. As none of them are able to preserve provenance, we design SPFlow. SPFlow is thus the only NonSP to SP rewriting approach able to preserve provenance, that is, able to maintain the semantics (meaning) of the workflow.

E Supervision of students and collaborations

This chapter presents my contributions to the problematic of workflow reuse. These contributions have been done in the context of two collaborations and involve two PhD students.

First, the work performed on workflow similarity search started in 2010 when Ulf Leser (from the Humboldt Universitat zu Berlin) spent a 6 months sabbatical at LRI to start working on scientific workflows. We then co-leaded a PHC Procope project in 2012[4]. In this collaborative context, Johannes Starlinger, a PhD student of Ulf leser that I co-supervised with him during two years, has developed the two main studies described in the first part of this Chapter. Johannes spent 3 months internship at the University of Pennsylvania, working with Susan Davidson and Sanjeev Khanna, where they started elaborate the very first hints for the Layer Decomposition approach. I have later contributed to this work by working with Johannes and with Susan Davidson when I visited UPenn in April 2014.

Second, the work on workflow rewriting to reduce the complexity of scientific workflows (SPFlow and DistillFlow) takes place within the context of the thesis of Jiuqiang Chen, that I have co-supervised with Christine Froidevaux. I have been invited in July 2012 at the University of Manchester after meeting Carole Goble at the “Principles of Provenance” Dagstuhl seminar. In Manchester, I met several workflow users and designers and I started considering the problem of redundancy in workflows. As a result, we have designed DistillFlow in collaboration with members of the Taverna workflow system (Carole Goble, Paolo Missier and Alan Williams).

F Conclusion and Perspectives

This chapter introduces my contributions to the domain of scientific workflow reuse.

[4]This project was co-funded by Campus France, formerly Egide, and DAAD, German Academic Exchange service

On the one hand, based on our preliminary study on workflow reuse [SCBL12], we present the comparison of a multitude of existing approaches to scientific workflow similarity search on the to-date largest human-curated corpus of similarity ratings for scientific workflows [SBCBL14]. We pay 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 shows that each step in the process of workflow comparison makes an important contribution to overall result quality. While, for practical reasons, our evaluation does 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.

We then introduce Layer Decompositon (LD) [SCBK+14], a novel approach for workflow comparison specifically tailored to measuring the similarity of scientific workflows. We comparatively evaluate 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. Considering runtime, we not only show our algorithm to be faster than other structure-aware approaches, but demonstrate how different algorithms can be combined to reduce the overall runtime while achieving comparable, or even improved result quality.

On the other hand, we introduce two approaches to make scientific workflow structures less complex. The main approach is dedicated to Taverna, DistillFlow [CBCG+14], and able to concretely remove redundancy in workflows by considering the concept of workflow anti-patterns. This approach has been driven by user needs expressed during interviews of end user, and evaluated on the workflows of the BioVel European project. The prototype [CCBF+14] is currently used by Taverna's users. The second approach is agnostic to the workflow system, SPFlow [CBFC12b], and able to rewrite a DAG workflow into a *provenance-equivalent* series-parallel workflow (in which complex graph operations such as subgraph isomorphism can be solved by polynomial algorithms). While the complexity of such algorithm is exponential, it provides reasonable results on real settings.

Perspectives in the domain of enhancing workflow reuse are numerous.

As mentioned in Chapter 2, provenance can play a major role: using provenance to retrieve workflows of interest allows considering similarity between input or output data which is probably the most important similarity feature to consider. Now that workflow similarity is better understood, various open questions can be considered in the domain of querying workflow repository and in particular the definition of a query language. In this direction, provenance query languages have been developed [SPG08, LLCF11, PA] notably based on the PROV standard (following OPM) [PD] while more formal frameworks and query languages, able to mix workflow and data provenance, have been introduced ([ADD+11, DODO+13]).

More generally, querying repositories of workflows (or workflow retrieval) may be addressed by a variety of solutions.

At one end of the spectrum are simple keyword queries that search a textual description of a workflow. This is the only type of search supported in current systems. At the other end of the spectrum, repositories should support full-fledged query languages encompassing predicates for searching IO-types of tasks, the topology of a workflow, keywords in the descriptions of tasks or the entire workflow, etc. [CBL11b, BZG$^+$15].

Largely unexplored are solutions in between and mixing textual and structural information. Our experience is that end users often know the type and format of data they provide (like Affymetrix GeneChip raw measurements, file format: CEL) and the type of data they expect as output (list of differentially expressed genes). Accordingly, they should be able to specify such parameters as constraints for a search, while the "intermediate" bits remain unspecified or only vaguely described. End users also may have a coarse-grained picture of the analysis they want to perform (like: "Data should be normalized, then aggregated by sample group, then analyzed for differential expression using a statistical test with a multiple testing correction") where only a few steps are already associated with concrete tools. Defining such *workflow sketches* to efficiently access workflow repositories is probably one of the most important challenges to cope with.

Last, generalizing the process followed by refactoring workflow approaches by possibly adding new anti-patterns and/or considering other workflow systems is another key point to reduce the complexity of workflows, and thus, to enhance workflow reuse.

Sommaire

CHAPTER 4

RANKING BIOLOGICAL DATA SETS USING CONSENSUS TECHNIQUES

Main Publications of the Chapter

[CBDH11] **Using Medians to generate consensus rankings for biological data**, SS-DBM 2011

[BRDCB14] **ConQuR-Bio: Consensus Ranking with Query Reformulation for Biological Data**, DILS 2014

This Chapter introduces my contributions to another important topic of data integration for the life science, namely, *biological data ranking*.

The aim of *biological data ranking* is to help users face with huge amount of data and choose between alternative pieces of information. This is of paramount importance in the context of querying data integration systems, where even very simple queries can return hundreds of answers. The need for ranking solutions, able to order answers is crucial for helping scientists organize their time and prioritize the new experiments to be possibly conducted. However, ranking biological data is a difficult task for various reasons: biological data are usually annotation files (e.g., a SwissProt, EntrezGene, or OMIM entry) which reflect expertise, they are thus associated with various degrees of *confidence* [CBBDF07]; data are not independent of each others' but they are linked by cross-references and the network formed by these links plays a role in the *popularity* of the data; the need expressed by scientists may also be taken into consideration whether the most well-known data should be ranked first, or the freshest, or the most surprising [HTL07]... As a consequence, although several ranking methods have been proposed in the last years within the bioinformatics community [BY06, SIY06, RWL$^+$06, HTL07], none of them has been deployed on systems currently used by the scientific community.

The work we present aims at addressing the problem that oncologists and pediatricians from the Institut Curie (Paris) and the Children's Hospital of Philadelphia (USA) face with, where too large numbers of answers are obtained to their biological queries.

To rank biological data, instead of combining ranking criteria and obtain a unique score for each data item, the general and original approach we propose to consider two steps. In the first step, several ranking methods are applied to biological data, independently of each others'. A given user query is considered and may be posed to one or several systems, and the results ordered using alternative ranking criteria and/or exploiting various ranking methods. In the second step, a consensus ranking approach is used to aggregate the input rankings.

To illustrate our approach, let us consider the following example, inspired from our concrete context of work, where an oncologist wants to know a set of genes involved in breast cancer. Depending on the database chosen and/or the ranking method to be used, several rankings of genes can be generated. In our example, let us consider the three following rankings: (i) the NCBI Entrez portal (the entry point to a set of major US biological databases) is queried and the genes obtained are sorted by "relevance" (where relevance is correlated to the number of times the keyword "breast cancer" appears in the annotation file of each gene) ; (ii) the SRS system (an entry point to a large set of biological databases) is queried and the results are ranked by freshness (i.e., more recent information first); (iii) the results obtained by Entrez are (loaded into a datawarehouse and) ranked using the PageRank algorithm. In such an example, three rankings, that is, three lists of genes, are obtained. In the second step, we propose to generate a *consensus ranking* to reflect the input rankings' common points while not putting too much importance on genes classified as "good" by only one or a few rankings (that is, minimizing their disagreements).

The problem of *consensus ranking*, also and mainly known as *rank aggregation*, has started to be investigated two centuries ago [dC85] by the Marquis de Condorcet to study voting systems. The problem has been actively studied again in the last two decades. Direct applications are numerous and include aggregating answers returned by several web engines [DKNS01], computing a global rating based on numerous user ratings [BK09, SBCBL14], determining the winner in a sport competition [BBN13], or combining biomedical orderings [CBDH11, DHF$^+$06, SM01]. This topic has been of particular interest in the information retrieval and database communities ([DKNS01] and [FKM$^+$04, FKM$^+$06, FKS03a, JKS14, SvZ09, SAYD$^+$13]) while several other communities have also deeply looked into it, including algorithmics ([Ail10, ACN08, SvZ09]) and datamining ([BGKN11]).

However, in real applications, the rankings to be aggregated may not be permutations where elements are strictly ordered, but they may have ties where some elements are placed at the same position (*e.g.*, several genes obtained as answer to a biological query may be considered as equally important). While the first efficient solution to rank aggregation considering input rankings with ties has been introduced by Fagin *et al.* in 2004 [FKM+04], most of the approaches and studies introduced since then have continued to focus on permutations, leaving several open questions in the context of ranking with ties.

Our contributions in this context are twofold.

First, we present a new heuristic for the problem, called *BioConsert* [CBDH11] (for generating Biological Consensus ranking with ties) that is dedicated to the features of biological data, in particular, that is able to fully consider consensus with ties.

Second, we present two applications we have considered for our consensus ranking methods. The first (and main) application uses consensus ranking for reconciling results obtained by alternative reformulations (using synonyms or equivalent terms in biomedical ontologies and thesaurus) of a biological query [BRDCB14]. And second, because ranking and scientific workflows are not independent topics of each other's, the second application has been done in the context of similarity search in scientific workflows where consensus ranking approaches help reconcile the experts ratings on the similarity of workflow pairs.

A BioConsert

In this section, we introduce the BioConsert heuristic which addresses the problem of consensus ranking and considers the features of the input rankings consisting of lists of data items (e.g., lists of genes). In this section we list the needs to be met by our solution, then we formalize the problem and introduce our solution while we demonstrate in the next section the interest of our approach in a real biological setting.

1 Requirements

Here, a ranking is represented by a suite of integers where each integer is one data object (in our case a gene provided as answer to a query).

Requirement 1: Comparing different sets. The distance to be chosen to compare rankings should take into account the fact that rankings are possibily over different sets of elements. Ranking are provided by systems which may play the role of filters. For instance, SRS or Entrez may not work on the exact same versions of the databases queried and/or may not interpret keyword queries the same way, resulting in different ranked data sets obtained to the same user query. In other words, two lists obtained by different ranking methods may not contain the same sets of data: it may be the case that ranking $R1$ mentions the data item #X which is not present in ranking $R2$. As a consequence, computing a distance between $R1$ and $R2$ implies taking into consideration those missing elements in rankings.

Requirement 2: Considering ties. The distance to be chosen to compare rankings should consider the fact that each ranking method output has the form of a list of sets (or ties) of answers. This may be the case when answers have groups of elements which are equally important. For instance, the following line $R1_{Q1} := [\{12, 21, 35, 36\}, \{41\}, \{3, 22\}]$ indicates that the ranking method $R1$ for the query $Q1$ returns 7 answers (genes in our example) and proposes a ranking in which 4 data objects are ranked equally first (#12,#21,#35,#36), strictly followed by the data object #41, itself strictly followed by the two data objects #3 and #22 (equally at the last position). In our setting, a set of genes may be considered as equally important to be involved in a disease because all genes of this set are involved in the same biological pathway.

Requirement 3: Minimizing disagreements. The consensus ranking we want to design should minimize the disagreements between rankings with ties. The distance to be considered should thus be able to penalize two kinds of disagreements: Besides considering as disagreements the classical cases where element i is ranked before element j in one ranking and after element j in the other, we need also to consider as disagreements cases where elements i, j are tied in one ranking (i.e. they are part of the same bucket such as #3 and #22 in the example above) but they are not tied in the other one.

In the following we introduce the preprocess we use on our data sets to go from rankings over different elements to rankings over the same sets of elements and we then introduce the distance we consider to meet these needs.

2 Unifying sets of data

We describe the steps of the *Unifying* preprocess to deal with different sets of data (Requirement 1) while minimizing disagreements between rankings (Requirement 3). Our aim is to penalize the fact that one element is considered in a ranking but not in another one.

1. Compute the union U of the elements (integers) appearing in each ranking.
2. For each ranking R_i, compute the set of elements contained in U but not in R_i, denoted here $U \setminus R_i$,
3. Augment each ranking the following way: add to R_i one new bucket at the latest position with elements from $U \setminus R_i$.

All the rankings obtained using the *Unifying* preprocess are thus over the same sets of elements. Additionally, if any ranking had elements that were not in the other rankings before these changes it will be penalized by the fact that this element will be ranked in the last bucket in all the other rankings.

3 Kendall-τ and generalized Kendall-τ distance

We can now choose a distance to compare rankings over the same sets of elements.

A good dissimilarity measure for comparing two rankings without ties is the Kendall-τ distance [Ken38] which counts the number of pairwise disagreements between positions of elements in those rankings. One way to generate a consensus permutation for a given set of permutations is to find a **median** for this set, that is, a permutation that minimizes the sum of Kendall-τ distances between this permutation and all permutations in the given set. The problem of finding the median of a set of m permutations of $\{1, 2, 3, \ldots, n\}$ under the Kendall-τ distance is a NP-hard problem (when $m \geq 4$, m even) that has been well-studied over the past years and for which good heuristics exist [ACN05, BFG+08, BCHV11, DKNS01, KMS07, vZW07].

Example 1.1 *In our context, the permutations are the rankings to be considered. Let* $r_1 = [5, 3, 2, 1, 4]$ *and* $r_2 = [1, 4, 5, 3, 2]$ *be two rankings over the data ids of the set* $\{1, 2, 3, 4, 5\}$. *Kendall-*τ *distance for*

*r_1 and r_2 is **12**, given by the following disagreements between pairs of elements of r_1 and r_2:*
1 appears **after** *2,3 and 5 in r_1 and* **before** *2, 3 and 5 in r_2*
2 appears **before** *1 and 4 in r_1 and* **after** *1 and 4 in r_2*
3 appears **before** *1 and 4 in r_1 and* **after** *1 and 4 in r_2*
4 appears **after** *2,3 and 5 in r_1 and* **before** *2, 3 and 5 in r_2*
5 appears **before** *1 and 4 in r_1 and* **before** *1 and 4 in r_2*

Following [FKM+04], a **bucket order** on $[n]$ is a transitive binary relation $\lhd$ for which there are non empty sets $\mathcal{B}_1, \ldots, \mathcal{B}_k$ (the **buckets**) that form a partition of $[n]$ such that $x \lhd y$ if and only if there are i, j with $i < j$ such that $x \in \mathcal{B}_i$ and $y \in \mathcal{B}_j$. Now, a **ranking with ties** is defined on $[n]$ as $R = [\mathcal{B}_1, \ldots \mathcal{B}_k]$, where $R[x] = i$ if $x \in \mathcal{B}_i$.

In this context, the **generalized Kendall-τ distance** can be defined analogously to the Kendall-τ distance while considering a parameter p such that $0 < p \leq 1$ to deal with the new kind of disagreements induced by ties. More precisely, the generalized Kendall-τ distance considers the number of disagreements between two rankings with ties (answering *Requirement 2*): a disagreement can be either two elements that are in different buckets in each ranking, where the order of the buckets disagree, and each such disagreement counts for 1 in the distance; or two elements that are in the same bucket in one ranking and in different buckets in the other, and each such disagreement counts for p (answering *Requirement 3*).

4 The BioConsert Heuristic

Given a set $\mathcal{R} = \{R_1, \ldots R_t\}$ of rankings with ties, the major concept of the BioConsert heuristic is to apply a series of "*good*" operations (namely, changeBucket and addBucket) on a starting ranking R_{start}, in order to make it closer to a median. We can then choose different rankings as possible start rankings and apply the BioConsert heuristic on each of them, keeping as **best consensus** the result of the best run.

The pseudo-code of the BioConsert algorithm is available in [CBDH11]. We have shown that BioConsert was able to handle biological data sets in a reasonable time and that it provided more accurate results (determined as the distance of the solution obtained by BioConsert to the exact solution, when available) than the current approximation algorithms of the literature (more information provided in the perspective Section of this Chapter).

B Applications of Consensus ranking approaches

1 Consensus of biological query reformulations

We now present the main application of BioConsert to address the needs expressed by our clinician collaborators. Instead of considering alternative ranking methods or alternative querying systems, we extend our approach and consider alternative reformulations of the user query. We place our solution on top of the Entrez NCBI querying system and provide an efficient and on-the-fly solution to the biological data ranking problem. We introduce ConQuR-Bio (Consensus ranking with Query Reformulation for Biological data), which makes use of two techniques: (i) query reformulation, where the user query is reformulated using various synonyms, where each reformulation provides one input ranking and (ii) consensus ranking, to exploit the complementarity of the results obtained by alternative reformulations.

1.1 General architecture

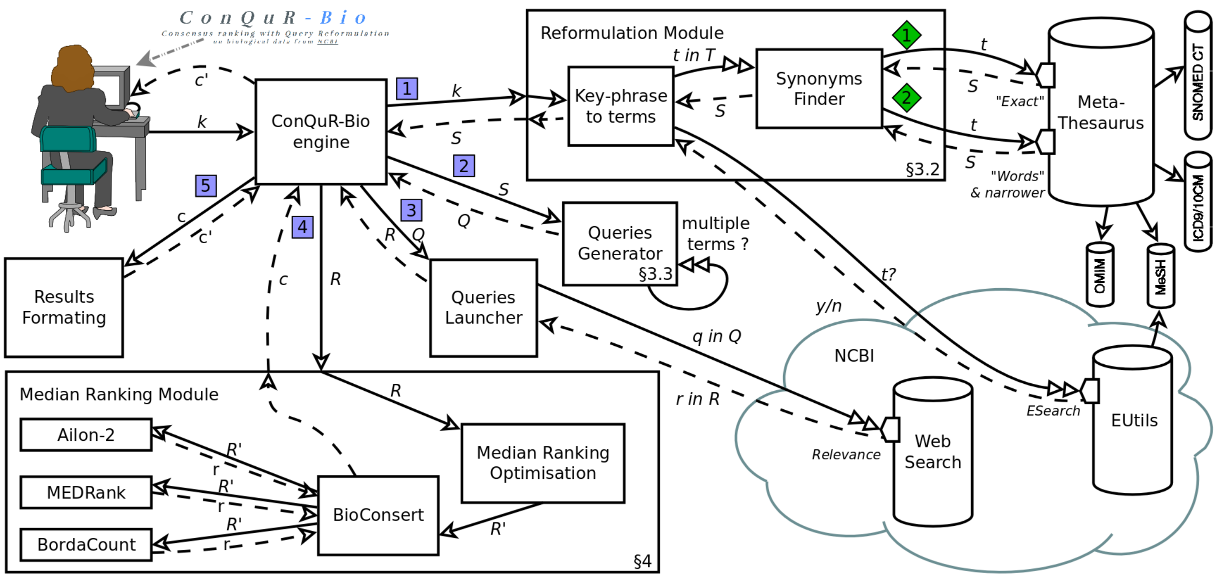


FIGURE 4.1 : Architecture of ConQuR-Bio

More precisely, as depicted in Figure 4.1, in ConQuR-Bio, the user provides a *key-phrase* k, consisting in a list of keywords. Several medical terminologies are then used to find synonyms for each keywords: The key-phrase is sent to the *Reformulation Module* which decomposes the key-phrase into a list T of MeSH (Medical Subject Headings) terms and leverage various terminologies within the UMLS® [Bod04] to generate the set S of synonyms. UMLS integrates more than 160 medical vocabularies including MeSH [Lip00], OMIM, but also SNOMED CT [SPSW01], a worldwide used clinical terminology often used as a core for Electronic Health Records, and ICD the International Classification of Diseases. Each UMLS concept is categorized with at least one *Semantic Type* (out of 150+) from the Semantic Network.

The set of synonyms S is then transmitted to the *Queries Generator* to be expressed as a set Q of queries to be run online on the NCBI search engine which provides sets of results ranked by *relevance*.

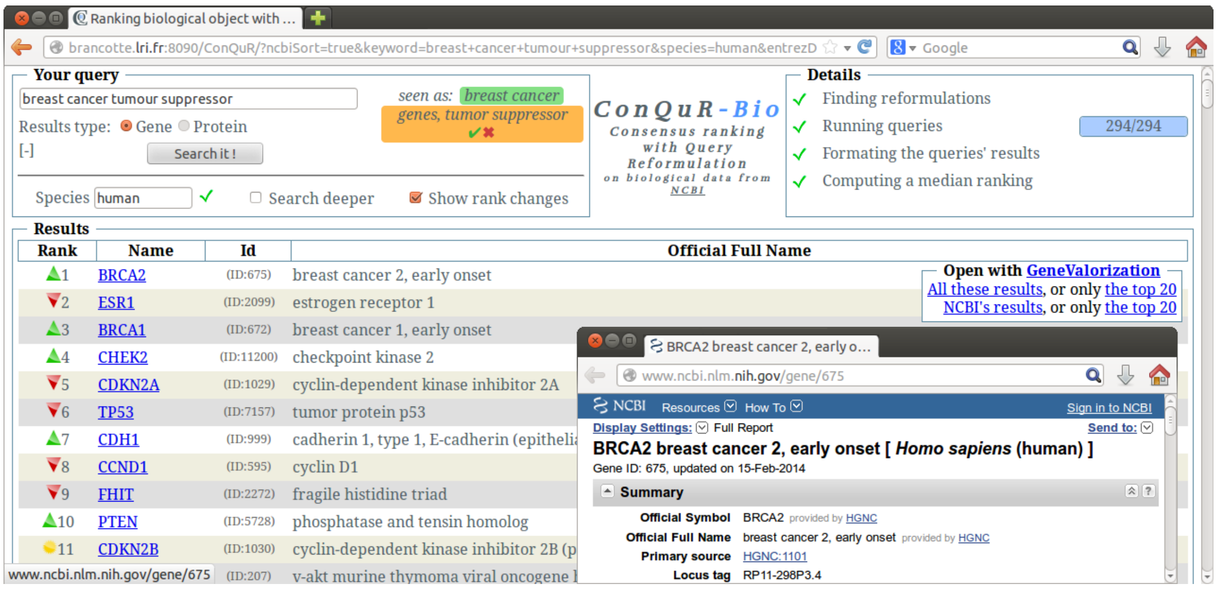
When all the ranked results R of queries Q have been collected, they are sent to the *Median Ranking Module* which is in charge of computing a unique consensus ranking. The BioConsert heuristics [CBDH11] is used and starts with results provided by three fast algorithms (BordaCount [Bor81], MEDRank [FKS03b], and Ailon's 2-approximation [Ail10]) to speed up the computation of the consensus.

Finally, the *Results Formatting* module enriches the ranking of gene identifiers with names and descriptions.

1.2 The ConQuR-Bio tool

The main interface of ConQuR-Bio is provided in Figure 4.2 and is composed of three areas, the query area in the top left panel, the running and progression details in the top right panel, and the results at the bottom.

In the query area, the key-phrase provided by the user is split into MeSH terms on-the-fly and dis-



Ranking biological object with ...

brancotte.lri.fr:8090/ConQuR/?ncbiSort=true&keyword=breast+cancer+tumour+suppressor&species=human&entrezD

Your query

breast cancer tumour suppressor

seen as: breast cancer genes, tumor suppressor ✓✗

Results type: [✓ Gene] Protein

[-] Search it !

Species human ✓ ☐ Search deeper [✓ Show rank changes]

ConQuR-Bio
Consensus ranking with Query Reformulation on biological data from NCBI

Details

✓ Finding reformulations
✓ Running queries 294/294
✓ Formating the queries' results
✓ Computing a median ranking

Results

Rank	Name	Id	Official Full Name
▲1	BRCA2	(ID:675)	breast cancer 2, early onset
▼2	ESR1	(ID:2099)	estrogen receptor 1
▲3	BRCA1	(ID:672)	breast cancer 1, early onset
▲4	CHEK2	(ID:11200)	checkpoint kinase 2
▼5	CDKN2A	(ID:1029)	cyclin-dependent kinase inhibitor 2A
▼6	TP53	(ID:7157)	tumor protein p53
▲7	CDH1	(ID:999)	cadherin 1, type 1, E-cadherin (epitheli
▼8	CCND1	(ID:595)	cyclin D1
▼9	FHIT	(ID:2272)	fragile histidine triad
▲10	PTEN	(ID:5728)	phosphatase and tensin homolog
●11	CDKN2B	(ID:1030)	cyclin-dependent kinase inhibitor 2B (p
		(ID:207)	v-akt murine thymoma viral oncogene

Open with GeneValorization
All these results, or only the top 20
NCBI's results, or only the top 20

www.ncbi.nlm.nih.gov/gene/675

BRCA2 breast cancer 2, early o...

www.ncbi.nlm.nih.gov/gene/675

NCBI Resources How To Sign in to NCBI

Display Settings: Full Report Send to:

BRCA2 breast cancer 2, early onset [*Homo sapiens* (human)]

Gene ID: 675, updated on 15-Feb-2014

Summary

Official Symbol	BRCA2 provided by HGNC
Official Full Name	breast cancer 2, early onset provided by HGNC
Primary source	HGNC:1101
Locus tag	RP11-298P3.4

FIGURE 4.2 : ConQuR-Bio interface and the window open after clicking on BRCA2.

played into colored boxes next to the key-phrase field. Colors indicate different status for a term: green when the term is recognized as a MeSH term, red when the term is not recognized, and orange when the term is matched to an existing MeSH term while the spelling is different. In addition to the orange semantic, when a term is matched to an alternative spelling, a check mark allows the user to accept the correction and update the key-phrase field, while a cross mark forces the system to use the given spelling.

The results area presents a ranking of genes with their official description as it can be found when browsing the NCBI website. Each gene is linked with its associated page in the NCBI Website, allowing the user to navigate in a familiar environment. In front of the rank of each gene, a symbol allows users to know whether the rank of the gene is raised, equal, lowered, or new in ConQuR-Bio compared to the results returned in the NCBI ranking.

Another interesting feature of ConQuR-Bio is its ability to provide the number of publications associated with each gene returned by calling the GeneValorization [BBBP^{+}11] tool, able to quickly browse PubMed.

1.3 Results on Biomedical queries

We now present the first results we obtained over a set of queries collected from collaborators of the *Institut Curie (France)* and the *Children's Hospital of Philadelphia (PA, USA)* and linked to their respective fields of expertise. The results presented considered 9 diseases: 7 cancers (*bladder, breast, cervical, colorectal, neuroblastoma, prostate, retinoblastoma*), one heart disease (the *Long QT Syndrome*), and one psychiatric disorder (the *attention deficit (with) hyperactivity disorder*). For cancers, we searched for information on the name of the cancer while also using additional words (and reformulations of such words) to refine the query, namely *tumor suppressor* and *oncogene*.

Evaluating such an approach is a difficult task as we face the users' perception of the results. In this very first series of experiments, we have chosen to focus on the first 20 results returned for each key-phrase (top-20).

Our clinician collaborators provide us with their *Gold standards*: the list l_d of the most relevant genes known to be associated with each disease d. The "goodness" of a consensus ranking c_d provided by ConQuR-Bio thus relies on the presence of elements of l_d in the top-ranked elements of c_d. In order to compare the results returned by ConQuR-Bio and the EntrezGene NCBI Web search engine with respect to gold standards, we used the Area Under the ROC Curve [Bra97] (ROC standing for *Receiver Operating Characteristic*) or *AUC* (closely related to precision and recall measures [Bra97]). The AUC aims at differentiating the presence of expected versus non expected data, taking into account the place of pieces of data (roughly, placing expected data before unexpected data increases the score of the AUC). AUC provides numbers ranged in $[0, 1]$, 1 being the highest score.

In Figure 4.3, we plot AUCs for the top-20 first results obtained for each key-phrase with both NCBI search engine and ConQuR-Bio. Globally, using ConQuR-Bio compared to NCBI allows to increase in average the AUC of 44.24%. More precisely, several points deserve attention.

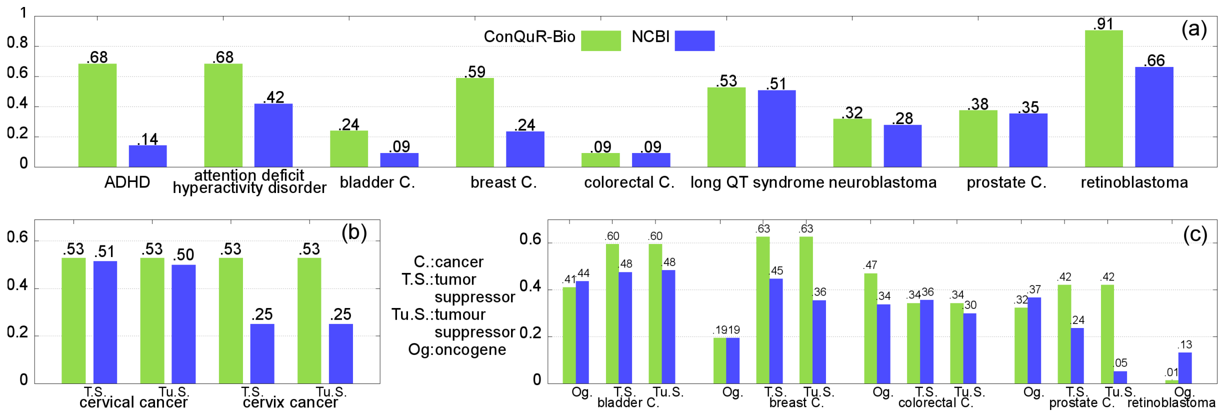


FIGURE 4.3 : The Area under the ROC curve (AUC) for the 20 first genes returned by ConQuR-Bio and the NCBI WebSearch for (a) Single-term key-phrases, (b) lexical variation around *cervix cancer tumor suppressor*, and (c) the remaining key-phrases.

- First, when focusing on single term key-phrases (i.e., considering the name of the disease only without adding *oncogene* or *tumo[u]r suppressor*, corresponding to Figure 4.3.a), ConQuR-Bio returns better results than the NCBI in 88.89% of the cases and always provides as good results as the NCBI. The average AUC is increased of 58.52% with ConQuR-Bio compared to NCBI.

- Second, multi-term key-phrases (Fig 4.3.b,c) have an AUC increased of 37.70% in average when using ConQuR-Bio compared to NCBI. This relatively less good result (37% vs. 58% of improvement) is actually due to the fact that the term *oncogene* has, in addition, one reformulation (*gene transforming*) less interesting (considered as "too vague" by our experts) than others.

- Third, considering *ADHD* and its unabbreviated name, the AUC is drastically increased using ConQuR-Bio. Also, as expected, the complete name and its abbreviation have different AUCs with the NCBI while remaining the same with ConQuR-Bio (since all the reformulations are considered). In the same spirit, lexical variations around the *cervical cancer tumor suppressor* (Fig 4.3.b) show the importance of taking into account all lexical and orthographic variations: ConQuR-Bio returns identical results for the four variants with an AUC of 0.53 while NCBI results have systematically inferior and variable AUCs.

- Finally, there were a few key-phrases, namely *colorectal cancer* and *neuroblastoma*, for which only plural reformulations were actually available (no actual synonyms available). The results obtained for such queries are then less impressive than in the previous cases while some of their respective AUCs are still increased compared to NCBI.

- A last point that deserves attention is the time taken by ConQuR-Bio to provide answers: While the NCBI search engine provides a ranking in at most $2s$, ConQuR-Bio takes $41s$ in average for the 9 single term key-phrases listed in Figure 4.3.a. This difference lies in the fact that the average number of synonyms retrieved by ConQuR-Bio (and thus the average number of queries to be answered and which elements should be ranked) is 17.

As a general conclusion, this first series of experiments appear to be very promising for using consensus ranking approaches to rank biological data.

We now briefly present another context of use of BioConsert.

2 Application to scientific workflows

As outlined in the previous Chapter, we have performed a user study where we have asked to fifteen experts to provide rating on workflow pairs. In this context, single experts' rankings are aggregated into consensus rankings using a variant of the BioConsert algorithm [CBDH11]. To take full advantage of BioConsert we extend it with the ability to deal with incomplete rankings using the *induced Kendall-τ* [DKNS01]. That is, if one user is *unsure* about a particular pair of workflows, then we omit that pair from his/her ranking. The ranking is then incomplete in the elements it contains. Typical consensus ranking algorithms either require input rankings to include the same elements, or append missing elements to the respective ranking (and thus introduce a bias in the produced consensus). Our extension of BioConsert with the *induced Kendall-τ distance* allows to produce its consensus directly from the incomplete lists. Besides, in case there exists more than one optimal consensus ranking, we average the algorithms' ranking performance over all variants.

C Related work

As stated in the introduction of this Chapter, the number of consensus ranking solutions is high [ACN05, BFG$^+$08, BCHV11, DKNS01, KMS07, vZW07]. There are only a few comparative studies on rank aggregation [BBN13, SvZ09, AM12] and although some conclusions may be drawn from them independently of each other's, the heterogeneity both in the data sets and in the set of algorithms considered makes the generalization of such results (i.e., their validity in other contexts) very difficult to assess. The need for designing a comparative experimental framework of such solutions and provide guidance to the user on which algorithm to follow is high.

As for the context of use, we have been pioneer in applying consensus ranking techniques to order biological data. First results on the biological significance of the results obtained are promising and will be deeper analyzed and discussed with our clinician collaborators.

D Supervision of students and collaborations

Faced with the amount of data returned by integration systems, the problem of biological data ranking is probably one of the most critical to address. Part of my PhD work addressed the problem of selecting and ranking paths between data sources to be followed to answer a given query. I have been interested in the problem of biological data ranking for years.

In 2009, as a master student under my supervision, Bryan Brancotte started concretely investigating the problem of biological data ranking: he tested the use of several existing ranking approaches (including variations of PageRank) to rank biological data.

In 2010, Sylvie Hamel, Professor in the University of Montréal spent a 4 month sabbatical in our group at Université Paris Sud. Sylvie Hamel worked on designing algorithms to construct consensus of variants of close genomes, where each genome is represented by an ordered set of genes. Her approach fit particularly well the problem of biological data ranking and I proposed to Sylvie Hamel to work on consensus rankings for biological data. We carefully formalized the new problem to be considered in collaboration with Alain Denise. The framework and associated problem were actually different from the initial setting: while genomes had the exact same set of genes in Sylvie Hamel's approach, rankings were not on the same set of elements, and while genes were ordered without equalities, we had to consider ties in the input data sets (not only permutations). As a first result of this collaboration, we published the original version of BioConsert at SSDBM 2011.

Bryan Brancotte then started his PhD thesis in 2012 under my co-supervision on the topic of *Consensus ranking approaches to order biological data*. ConQuR-Bio is at the center of his PhD work and involves a collaboration with Bastien Rance, expert in biological terminologies and ontologies in medical contexts, currently working at Hopital Europeen Georges Pompidou (and previously postdoc at the National Institute of Health). Bryan also actively worked in designing and implementing a comparison framework for consensus approaches (under revision at VLDB 2015, see the perspectives below).

E Conclusion

This Chapter introduces my contributions in the context of biological data ranking.

We design and implement a new method providing a consensus from rankings possibly composed of different elements, with lists of sets (ties) while minimizing the disagreements between rankings. More precisely, we propose a preprocess able to deal with different data sets to then be able to work on rankings with ties on the same data sets. Then, while the general problem of finding a median of a set of rankings with ties is NP hard, we introduce the BioConsert [CBDH11] heuristic. Comparison with a large set of approximation algorithms and other heuristics algorithms currently available in the literature demonstrates the ability of BioConsert to efficiently deal with real, large data sets while providing high quality results.

With ConQuR-Bio [BRDCB14], we make the connection between the *query reformulation* field and the *median ranking* field. We leverage terminologies integration in the UMLS system (an approach and system shown to be effective [DFAJY$^+$11]) to propose reformulations. We provide reformulations based on MeSH terms identified in the users key-phrases. To generate a consensus answer to the user emphasizing the agreements between the reformulations, we backe its computation on BioConsert combined with several median ranking algorithms, allowing the system to quickly compute a consensus. We compare our approach to the main portal used to browse gene-centric biological data, namely the EntrezGene database from the NCBI website and its ranking function based on relevance. We show that when measuring the presence and order of expected results (based on gold standards), ConQuR-Bio outperforms the NCBI. Last but not least, the system is available and free to use at `http://conqur-bio.lri.fr` as a website.

The Chapter ends with another illustration of the application of consensus algorithms, in the context of scientific workflows, to reconcile expert's point of views on workflow similarity pairs.

The central question in our ongoing work is on the commonalities and differences between existing consensus ranking algorithms. A paper on this direction is currently under revision at VLDB 2015 [BYB+15]. In such a paper, we propose a classification of the approaches and consider data features (e.g., similarity between input rankings, size of the data sets) and other features from the context of use (e.g., number of input rankings considered) to be exploited to guide the user in the choice of an algorithm. We evaluate all approaches on very large and various data sets, both real and carefully generated using random generation procedures.

The validation of the use of consensus ranking for biological data from a more qualitative point-of-view, in close collaboration with our experts is another part of future work. Besides, ConQuR-Bio can be improved following several directions for example by exploring the detection of concepts from the users key-phrases by deploying concept recognition software which are well established in the bioinformatics community such as MetaMap [Aro01] or the BioAnnotator tool, enabling advanced reformulation options (e.g. different levels of granularity).

Besides, a large number of more theoretical points on consensus ranking are also challenging. As the complexity of computing a median (exact solution) of m input permutations under the Kendall-tau distance is only known to be NP-hard for $m \geq 4$ permutations, m even, the central open problem in rank aggregation is to know the complexity of the problem when m is odd. Complexity of the problem is also open when considering ties. Another point to be investigated is to identify polynomial-time solvable cases of the problem at hand that are obtained by imposing restrictions on the possible inputs based on the data features and context of use introduced above. From a fixed-parameter tractability point-of-view, providing a broad study of the parameterized complexity for computing optimal Kemeny rankings (i.e., medians) would be particularly valuating. In particular, it would help to further identify meaningful different parameterizations and combinations of single parameters and their impact on problem complexity.

Sommaire

Chapter 5 Perspectives

This thesis has presented several of my main contributions obtained in the last 9 years to the domain of data integration in the life sciences, both on managing and querying scientific workflows, including workflow provenance, similarity and refactoring aspects, and on consensus ranking for biological data. Contributions have been obtained both in computer science and bioinformatics point-of-views and have systematically involved close collaborations with biologists or clinicians. Our ongoing work includes designing query languages for workflow repositories, exploiting and representing provenance and defining a comparative framework for consensus ranking to provide guidance to users on which approach to favor.

More generally, challenges in each of these domains are still numerous. Scientific workflows have now reached a level of maturity making them able to deal with large-scale amounts of complex data in production, opening the door to several open research questions directly related to the big data paradigm: how to store, index, query and efficiently analyze the huge and highly distributed amounts of data concretely produced by in-silico experiments? Advances in managing scientific workflows depend – and may have impact on – progress made in other communities such as system biology or social networks or more generally algorithmics and graphs. As for consensus approaches, several complexity questions are key points to be addressed in the future, considering rankings with ties, understanding salient data features to design parametric complexity solutions.

Since scientific workflow systems are increasingly equipped with a set of functionalities which fit particularly well with the management of complex and large-scale bioinformatics experiments, targeting a wider user community, including scientists which are currently not using any workflow management system to design and execute their experiments, is of paramount importance. The main two research directions I am interested in following in the next three to five years are directly related to this point.

I base the first line of research on one main observation obtained for user interviews: users would be more inclined to share and upload their workflows into repositories if they were sure that reused workflows are correctly accompanied with citations acknowledging their work. I thus want to address the problem of *workflow citation* by considering both current repositories (reconstruct the history of existing workflows) and designing dedicated citation modules for repositories (construct the history of new workflows and maintain it).

Besides, scripts (in Perl, Python, Java...) and workflow design are the current alternative two means offered to users to implement a biological analysis. My second line of research aims to bridge the gap between those two paradigms.

BIBLIOGRAPHY

[ABGK13] P. APPEL, K. BELHAJJAME, C. GOBLE, and P. KARAGOZ. « Small Is Beautiful: Summarizing Scientific Workflows Using Semantic Annotations ». In *IEEE 2nd International Congress on Big Data*, 2013. 20

[ABL09] MK ANAND, S BOWERS, and B LUDÄSCHER. « A navigation model for exploring scientific workflow provenance graphs ». In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, Page 2. ACM, 2009. 20

[ABL10] M K. ANAND, S. BOWERS, and B. LUDÄSCHER. « Techniques for efficiently querying scientific workflow provenance graphs. ». *International Conference on Extending Database Technology*, 10:287–298, 2010. 20

[ABML09] MK. ANAND, S. BOWERS, T. MCPHILLIPS, and B. LUDÄSCHER. « Efficient provenance storage over nested data collections ». In *Proceedings of the International Conference on Extending Database Technology*, pp. 958–969. ACM, 2009. 20

[ACN05] N. AILON, M. CHARIKAR, and N. NEWMAN. « Aggregating inconsistent information: Ranking and clustering ». In *Proc. of the Symposium on the Theory of Computing*, pp. 684–693, New York, NY, USA, 2005. ACM. 50, 55

[ACN08] N. AILON, M. CHARIKAR, and A. NEWMAN. « Aggregating inconsistent information: ranking and clustering ». *Journal of the ACM (JACM)*, 55(5):23, 2008. 48

[ADD$^+$11] Y. AMSTERDAMER, S.B DAVIDSON, D. DEUTCH, T. MILO, J. STOYANOVICH, and V. TANNEN. « Putting lipstick on pig: Enabling database-style workflow provenance ». *VLDB, Proceedings of the International Conference on Very Large Data Bases*, 5(4):346–357, 2011. . . . . 23, 45

[AFFW13] G ALOISIOA, S FIOREA, Ian FOSTER, and D WILLIAMS. « Scientific big data analytics challenges at large scale ». *Proceedings of Big Data and Extreme-scale Computing (BDEC)*, 2013. 23

[AG97] A. APOSTOLICO and Z. GALIL, editors. *Pattern matching algorithms*. Oxford University Press, Oxford, UK, 1997. 21

[Ail10] N. AILON. « Aggregation of partial rankings, p-ratings and top-m lists ». *Algorithmica*, 57(2):284–300, 2010. . . . . 48, 52

[AM12] A. ALI and M. MEILĂ. « Experiments with Kemeny ranking: What works when? ». *Mathematical Social Sciences*, 64(1):28–40, 2012. 55

[AMA06] A. AKRAM, D. MEREDITH, and R. ALLAN. « Evaluation of BPEL to scientific workflows ». In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, vol. 1, pp. 269–274. IEEE, 2006. 3

[Aro01] Alan R ARONSON. « Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. ». In *Proceedings of the AMIA Symposium*, Page 17. American Medical Informatics Association, 2001. 57

[BBBP$^+$11] B. BRANCOTTE, A. BITON, I. BERNARD-PIERROT, F. RADVANYI, F. REYAL, and S. COHEN-BOULAKIA. « Gene List significance at-a-glance with GeneValorization ». *Bioinformatics*, 27(8):1187–1189, 2011. 53

[BBN13] N. BETZLER, R. BREDERECK, and R. NIEDERMEIER. « Theoretical and empirical evaluation of data reduction for exact Kemeny Rank Aggregation ». *Autonomous Agents and Multi-Agent Systems*, pp. 1–28, 2013. . . . . 48, 55

[BCBD07] O. BITON, S. COHEN-BOULAKIA, and S. B. DAVIDSON. « Zoom*UserViews: Querying Relevant Provenance in Workflow Systems ». In *VLDB, Proc. of the International Conference on Very Large Data Bases*, pp. 1366–1369. ACM, 2007. . . . . . 4, 7, 12, 22

[BCBD$^+$08] Z. BAO, S. COHEN-BOULAKIA, S.B. DAVIDSON, A. EYAL, and S. KHANNA. « Algorithms for Differencing Workflow Runs ». Internal report MS-CIS-08-04, University of Pennsylvania, 2008. 17

[BCBD$^+$09] Z. BAO, S. COHEN-BOULAKIA, S.B. DAVIDSON, A. EYAL, and S. KHANNA. « Differencing Provenance in Scientific Workflows ». In *Proc. of the Int. Conf. on Data Engineering (ICDE), IEEE*, pp. 808–819, 2009. . . . . 4, 7, 18, 19, 22

[BCBDG09] Z. BAO, S. COHEN-BOULAKIA, S. DAVIDSON, and P. GIRARD. « PDiffView: Viewing the Difference in Provenance of Workflow Results ». *VLDB, Proc. of the Int. Conf. on Very Large Data Bases*, 2(2):1638–1641, 2009. . . . . 5, 7, 22

[BCBDH08] O. BITON, S. COHEN-BOULAKIA, S.B. DAVIDSON, and C.S. HARA. « Querying and Managing Provenance through User Views in Scientific Workflows ». In *Proc. of the Int. Conf. on Data Engineering (ICDE), IEEE*, pp. 1072–1081, 2008. 4, 7, 12, 15, 22, 41

[BCC06] P. BUNEMAN, A. CHAPMAN, and J. CHENEY. « Provenance Management in Curated Databases ». In *ACM SIGMOD international conference on Management of data*, pp. 539–550, 2006. 8

[BCHV11] G. BLIN, M. CROCHEMORE, S. HAMEL, and S. VIALETTE. « Medians of an odd number of permutations ». *Pure Mathematics and Applications*, 21-2:161–175, 2011. 50, 55

[BCTV04] D. Bhagwat, L. Chiticariu, W.C. Tan, and G. Vijayvargiya. « An Annotation Management System for Relational Databases ». In *VLDB, Proceedings of the International Conference on Very Large Data Bases*, pp. 900–911, 2004. 8

[BDKR09] O Biton, S B. Davidson, S Khanna, and S Roy. « Optimizing user views for workflows ». In *Proc. of the International Conference on Database Theory*, pp. 310–323, 2009. . . . . 15, 20

[BFG$^+$08] N. Betzler, M.R. Fellows, J. Guo, R. Niedermeier, and F.A Rosamond. « Fixed-Parameter Algorithms for Kemeny Scores ». In *Proceedings of the 4th Algorithmic Aspects in Information and Management*, LNCS 5034, pp. 60–71. Springer, 2008. 50, 55

[BG12] R. Bergmann and Y. Gil. « Similarity assessment and efficient retrieval of semantic workflows. ». *Information Systems*, 40:115–127, 2012. . . . . 31, 43

[BGKN11] N. Betzler, J. Guo, Ch. Komusiewicz, and R. Niedermeier. « Average parameterization and partial kernelization for computing medians ». *Journal of Computer and System Sciences*, 77(4):774–789, 2011. 48

[BK09] J.P. Baskin and S. Krishnamurthi. « Preference aggregation in group recommender systems for committee decision-making ». In *Proceedings of the third ACM conference on Recommender systems*, pp. 337–340. ACM, 2009. 48

[BKS92] WW. Bein, J Kamburowski, and MF. M. Stallmann. « Optimal Reductions of Two-Terminal Directed Acyclic Graphs ». *SIAM J. Comput.*, 21(6):1112–1129, 1992. 41, 44

[BKT01] P. Buneman, S. Khanna, and W. Tan. « Why and Where: A Characterization of Data Provenance. ». In *Proc. of the International Conference on Database Theory*, pp. 316–330, 2001. 8

[BL05] S. Bowers and B. Ludäscher. « Actor-Oriented Design of Scientific Workflows ». In *Int. Conf. on Concept. Modeling*, pp. 369–384, 2005. . . . . 12, 20

[BL13] M. Bux and U. Leser. « Parallelization in scientific workflow management systems ». *arXiv preprint arXiv:1303.7195*, 2013. 23

[BML$^+$06] S. Bowers, T.M. McPhillips, B. Ludäscher, S. Cohen, and S.B. Davidson. « A Model for User-Oriented Data Provenance in Pipelined Scientific Workflows. ». In *International Provenance and Annotation Workshop (IPAW)*, vol. 4145 de *LNCS*, pp. 133–147. Springer, 2006. 8

[BNL$^+$04] J. Bleiholder, F. Naumann, Z. Lacroix, L. Raschid, H. Murthy, and M.-E. Vidal. « BioFast: challenges in exploring linked life sciences sources ». *ACM SIGMOD Record*, 33(2):72–77, 2004. 6

[Bod04] O. Bodenreider. « The unified medical language system (UMLS): integrating biomedical terminology ». *Nucleic acids research*, 32(suppl 1):D267–D270, 2004. 52

[Bor81] J.C.de Borda. « Mémoire sur les élections au scrutin ». *Histoire de l'academie royal des sciences*, pp. 657 – 664, 1781. 52

[bpe] « Business Process Execution Language for Web Services ». http://www.ibm.com/developerworks/library/specification/ws-bpel/. 16

[Bra97] A.P. BRADLEY. « The use of the area under the ROC curve in the evaluation of machine learning algorithms ». *Pattern Recognition*, 30:1145–1159, 1997. 54

[BRDCB14] B. BRANCOTTE, B. RANCE, A. DENISE, and S. COHEN-BOULAKIA. ConQuR-Bio: Consensus Ranking with Query Reformulation for Biological Data. In *Data Integration in the Life Sciences*, vol. 8574 de *LNCS*, pp. 128–142. 2014. . . . . 6, 47, 49, 56

[BSHW06] O. BENJELLOUN, A. Das SARMA, A. HALEVY, and J. WIDOM. « ULDBs: Databases with Uncertainty and Lineage. ». In *VLDB, Proceedings of the International Conference on Very Large Data Bases*, pp. 953–964, 2006. . . . . 8, 12

[BSM$^+$08] BT BLAUSTEIN, L SELIGMAN, M MORSE, M D ALLEN, and A ROSENTHAL. « PLUS: Synthesizing privacy, lineage, uncertainty and security. ». In *ICDE Workshops*, pp. 242–245, 2008. 20

[BTN$^+$10] J. BHAGAT, F. TANOH, E. NZUOBONTANE, T. LAURENT, J. ORLOWSKI, M. ROOS, K. WOLSTENCROFT, S. ALEKSEJEVS, R. STEVENS, S. PETTIFER, and OTHERS. « BioCatalogue: a universal catalogue of web services for the life sciences ». *Nucleic acids research*, Page 394, 2010. 3

[BY06] A. BIRKLAND and G. YONA. « BIOZON: a system for unification, management and analysis of heterogeneous biological data ». *BMC Bioinformatics*, 7(1):70, 2006. 6, 48

[BYB$^+$15] B. BRANCOTTE, B. YANG, G. BLIN, S. COHEN-BOULAKIA, A. DENISE, and S. HAMEL. « Rank aggregation with ties: experiments and analysis. ». *Submitted*, 2015. 57

[BZG$^+$15] K BELHAJJAME, J ZHAO, D GARIJO, M GAMBLE, K HETTNE, R PALMA, E MINA, O CORCHO, J GÓMEZ-PÉREZ, S BECHHOFER, and OTHERS. « Using a suite of ontologies for preserving workflow-centric research objects ». *Web Semantics: Science, Services and Agents on the World Wide Web*, 2015. 46

[CBBCD08] S. COHEN-BOULAKIA, O. BITON, S. COHEN, and Susan B. DAVIDSON. « Addressing the Provenance Challenge using ZOOM ». *Concurrency and Computation: Practice and Experience*, Vol 20(5):497–506, 2008. . . . . 4, 7, 12, 13, 22

[CBBDF07] S. COHEN-BOULAKIA, O. BITON, S. DAVIDSON, and Ch. FROIDEVAUX. « BioGuideSRS: querying multiple sources with a user-centric perspective ». *Bioinformatics*, 23(10):1301–1303, 2007. 48

[CBCG$^+$14] S. COHEN-BOULAKIA, J. CHEN, C. GOBLE, P. MISSIER, A. WILLIAMS, and Ch. FROIDEVAUX. « Distilling Structure in Taverna Scientific Workflows: A refactoring approach. ». *BMC Bioinformatics*, 15(1):S12, 2014. . . . . 5, 25, 45

[CBDF$^+$06] S COHEN-BOULAKIA, S DAVIDSON, Ch FROIDEVAUX, Z LACROIX, and M-E VIDAL. « Path-based systems to guide scientists in the maze of biological data sources ». *Journal of Bioinformatics and Computational Biology*, 4(05):1069–1095, 2006. . 6

[CBDH11] S. COHEN-BOULAKIA, A. DENISE, and S. HAMEL. « Using medians to generate consensus rankings for biological data ». In *Scientific and Statistical Database Management*, vol. 6809 de *LNCS*, pp. 73–90. Springer, 2011. 6, 32, 47, 48, 49, 51, 52, 55, 56

[CBFC12a] S. COHEN-BOULAKIA, Ch. FROIDEVAUX, and J. CHEN. « Formal aspects of SPFlow (Proofs, complexity) ». In *Internal report, LRI, University Paris Sud, http : //www.lri.fr/˜cohen/Spflow − extended.pdf*, 2012. 39

[CBFC12b] S. COHEN-BOULAKIA, Ch. FROIDEVAUX, and J. CHEN. « Scientific workflow rewriting while preserving provenance ». In *Proc. of IEEE Int. Conf. on E-Science (e-Science)*, pp. 1–9. IEEE Computer Society, 2012. . . . . 5, 25, 42, 45

[CBL11a] S. COHEN-BOULAKIA and U. LESER. « Next generation data integration for Life Sciences ». In *Proc. of the Int. Conf. on Data Engineering (ICDE), IEEE*, pp. 1366–1369, 2011. 1

[CBL11b] S. COHEN-BOULAKIA and U. LESER. « Search, adapt, and reuse: the future of scientific workflows ». *SIGMOD Record*, 40(2):6–16, 2011. . . . . 1, 3, 46

[CBLS$^+$04] S. COHEN-BOULAKIA, S. LAIR, N. STRANSKY, S. GRAZIANI, F. RADVANYI, E. BARILLOT, and C. FROIDEVAUX. « Selecting Biomedical Data Sources according to User Preferences ». *Bioinformatics*, 20:i86–i93, 2004. 6

[CBT09] S. COHEN-BOULAKIA and WC. TAN. Provenance in Scientific Databases. In *Encyclopedia of Database Systems*, pp. 2202–2207. Springer US, 2009. 7

[CCBD06] S. COHEN, S. COHEN-BOULAKIA, and S. DAVIDSON. « Towards a Model of Provenance and User Views in Scientific Workflows ». In *Data Integration in the Life Sciences*, vol. 4075 de *LNBI*, pp. 264–279. Springer, 2006. . . . . 4, 12

[CCBF$^+$14] J. CHEN, S. COHEN-BOULAKIA, Ch. FROIDEVAUX, C. GOBLE, P. MISSIER, and A. WILLIAMS. « DistillFlow: Removing redundancy in Scientific Workflows ». In *Proc. of International Conference on Scientific and Statistical Database Management*, 2014. . . . . 5, 25, 45

[CCL$^+$08] A. CHEBOTKO, S. CHANG, S. LU, F. FOTOUHI, and Ping YANG. « Scientific workflow provenance querying with security views ». In *Web-Age Information Management, 2008. WAIM'08. The Ninth International Conference on*, pp. 349–356. IEEE, 2008. 20

[CCPR13] Y. CHEAH, R. CANON, B. PLALE, and L. RAMAKRISHNAN. « Milieu: Lightweight and configurable big data provenance for science ». In *Big Data (BigData Congress), 2013 IEEE International Congress on*, pp. 46–53. IEEE, 2013. 23

[CGHX06] K CHAKRABARTI, V GANTI, J HAN, and D XIN. « Ranking objects based on relationships ». In *ACM SIGMOD international conference on Management of data*, pp. 371–382. ACM, 2006. 6

[Che01] W. CHEN. « New Algorithm for Ordered Tree-to-Tree Correction Problem. ». *J. Algorithms*, 40(2):135–158, 2001. 21

[Cle14] D. CLEMENTS. « Scaling Galaxy for Big Data ». *EMBnet. journal*, 20(A):e764, 2014. 22

[COM14] F. COSTA, D. de OLIVEIRA, and M. MATTOSO. « Towards an Adaptive and Distributed Architecture for Managing Workflow Provenance Data ». In *e-Science (e-Science), 2014 IEEE 10th International Conference on*, vol. 2, pp. 79–82. IEEE, 2014. 23

[COO$^+$10] F. COSTA, D. de OLIVEIRA, E. OGASAWARA, AAB. LIMA, and M. MATTOSO. « Athena: Text Mining Based Discovery of Scientific Workflows in Disperse Repositories ». *RED*, pp. 104–121, 2010. . . . . 31, 43

[CRGMW96] S.S. CHAWATHE, A. RAJARAMAN, H. GARCIA-MOLINA, and J. WIDOM. « Change Detection in Hierarchically Structured Information. ». In *ACM SIGMOD international conference on Management of data*, pp. 493–504, 1996. 21

[CSF13] F.S. CHIRIGATI, D. SHASHA, and J. FREIRE. « ReproZip: Using Provenance to Support Computational Reproducibility. ». In *TaPP, Theory and Practice of Provenance*, 2013. 22

[CW01] Y. CUI and J. WIDOM. « Lineage Tracing for General Data Warehouse Transformations ». In *VLDB, Proceedings of the International Conference on Very Large Data Bases*, pp. 471–480, 2001. 8

[dC85] Marquis de CONDORCET. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Imprimerie Royale, 1785. 48

[DCBE$^+$07] S.B. DAVIDSON, S. COHEN-BOULAKIA, A. EYAL, B. LUDÄSCHER, T.M. MCPHILLIPS, S. BOWERS, M. ANAND, and J. FREIRE. « Provenance in Scientific Workflow Systems ». *IEEE Data Eng. Bull.*, 30(4):44–50, 2007. 7

[DF08] S.B. DAVIDSON and J. FREIRE. « Provenance and scientific workflows: challenges and opportunities ». In *Proceedings of the ACM SIGMOD international conference on Management of data*, pp. 1345–1350. ACM, 2008. 4

[DFAJY$^+$11] D. DEMNER-FUSHMAN, S. ABHYANKAR, A. JIMENO-YEPES, R.F LOANE, B. RANCE, F. LANG, N.C IDE, E. APOSTOLOVA, and AR ARONSON. « A Knowledge-Based Approach to Medical Records Retrieval. ». In *TREC*, 2011. . . . 56

[DHF$^+$06] R.P DECONDE, S. HAWLEY, S. FALCON, N. CLEGG, B. KNUDSEN, and R. ETZIONI. « Combining results of microarray experiments: a rank aggregation approach ». *Statistical Applications in Genetics and Molecular Biology*, 5(1), 2006. 48

[DKBL12] SC DEY, S. KÖHLER, S. BOWERS, and B. LUDÄSCHER. « Datalog as a Lingua Franca for Provenance Querying and Reasoning. ». In *TaPP, Theory and Practice of Provenance*, 2012. 23

[DKNS01] C. DWORK, R. KUMAR, M. NAOR, and D. SIVAKUMAR. « Rank Aggregation Methods for the Web ». In *Proc. of the 10th World Wide Web conference*, pp. 613–622. ACM, 2001. . . . . 48, 50, 55

[DKR$^+$11] SB DAVIDSON, S KHANNA, S ROY, J STOYANOVICH, Val TANNEN, and Yi CHEN. « On provenance and privacy ». In *Proceedings of the 14th International Conference on Database Theory*, pp. 3–10. ACM, 2011. 20

[DODO$^+$13] J DIAS, E OGASAWARA, D DE OLIVEIRA, F PORTO, P VALDURIEZ, and M MATTOSO. « Algebraic dataflows for big data analysis ». In *Big Data, 2013 IEEE International Conference on*, pp. 150–155. IEEE, 2013. 45

[dOOBM12] D. de OLIVEIRA, K. ACS OCAÑA, F. BAIÃO, and M. MATTOSO. « A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds ». *Journal of Grid Computing*, 10(3):521–552, 2012. 23

[DZG$^+$12] Y DEMCHENKO, Z ZHAO, P GROSSO, A WIBISONO, and C de LAAT. « Addressing Big Data challenges for Scientific Data Infrastructure. ». In *CloudCom*, pp. 614–617, 2012. 23

[FKM$^+$04] R. FAGIN, R. KUMAR, M. MAHDIAN, D. SIVAKUMAR, and E. VEE. « Comparing and aggregating rankings with ties ». In *Proc. of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 47–58. ACM, 2004. . . 48, 49, 51

[FKM$^+$06] R. FAGIN, R. KUMAR, M. MAHDIAN, D SIVAKUMAR, and E. VEE. « Comparing partial rankings ». *SIAM Journal on Discrete Mathematics*, 20(3):628–648, 2006. . 48

[FKS03a] R. FAGIN, R. KUMAR, and D. SIVAKUMAR. « Comparing top k lists ». *SIAM Journal on Discrete Mathematics*, 17(1):134–160, 2003. 48

[FKS03b] R. FAGIN, R. KUMAR, and D. SIVAKUMAR. « Efficient similarity search and classification via rank aggregation ». In *Proc. of the ACM SIGMOD int. conf. on Management of data*, pp. 301–312. ACM, 2003. 52

[FR10] N. FRIESEN and S. RÜPING. « Workflow Analysis Using Graph Kernels ». *SoKD*, 2010. . . . . 31, 43

[FSC$^+$06] J. FREIRE, C.T. SILVA, S.P. CALLAHAN, E. SANTOS, C.E. SCHEIDEGGER, and H.T. VO. « Managing Rapidly-Evolving Scientific Workflows. ». *International Provenance and Annotation Workshop (IPAW)*, pp. 10–18, 2006. . . . . 8, 20

[FVWZ02] I. FOSTER, J. VOCKLER, M. WILDE, and Y. ZHAO. « Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation ». In *Proc. of International Conference on Scientific and Statistical Database Management*, pp. 37–46, 2002. 8

[GAB$^+$12] D GARIJO, P ALPER, K BELHAJJAME, Ó CORCHO, Y GIL, and CA GOBLE. « Common motifs in scientific workflows: An empirical analysis ». In *Proc. of the 8th IEEE International Conference on E-Science, e-Science*. IEEE Computer Society, 2012. 44

[GE03] A. GONZÁLEZ-ESCRIBANO. « *Synchronization Architecture in Parallel Programming Models* ». PhD thesis, University of Valladolid, 2003. 44

[GKT07] TJ. GREEN, G KARVOUNARAKIS, and V TANNEN. « Provenance semirings ». In *Principles of Database Systems (PODS)*, pp. 31–40, 2007. 8

[GLG06] A. GODERIS, P. LI, and C. GOBLE. « Workflow discovery: the problem, a case study from e-Science and a graph-based solution ». *ICWS*, pp. 312–319, 2006. . . . . . 31, 43

[GNT10] 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. 22

[GS08] C GOBLE and R STEVENS. « State of the nation in data integration for bioinformatics ». *Journal of biomedical informatics*, 41(5):687–693, 2008. 2

[Har87] D. HAREL. « Statecharts: A visual formalism for complex systems. ». *Science of Comp. Programming*, 8:231–274, 1987. 20

[HTL07] P. HUSSELS, S. TRISSL, and U. LESER. « What's new? What's certain?–Scoring Search Results in the Presence of Overlapping Data Sources ». In *Data Integration in the Life Sciences*, pp. 231–246. Springer, 2007. . . . . 6, 48

[Ive09] ZG IVES. « Data integration and exchange for scientific collaboration ». In *Data Integration in the Life Sciences*, pp. 1–4. Springer, 2009. 6

[JKS14] M. JACOB, B. KIMELFELD, and J. STOYANOVICH. « A System for Management and Analysis of Preference Data ». *Proc. of the VLDB Endowment*, 7(12), 2014. 48

[Ken38] M. KENDALL. « A New Measure of Rank Correlation ». *Biometrika*, 30:81–89, 1938. 50

[KGH$^+$12] N. KONG, T. GROSSMAN, B. HARTMANN, M. AGRAWALA, and G. FITZMAURICE. « Delta: a tool for representing and comparing workflows ». In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1027–1036. ACM, 2012. 20

[Kle98] P.N. KLEIN. « Computing the Edit-Distance between Unrooted Ordered Trees ». In *ESA '98: Proceedings of the 6th Annual European Symposium on Algorithms*, pp. 91–102, 1998. 21

[KMS07] C. KENYON-MATHIEU and W. SCHUDY. « How to rank with few errors ». In *Proc. of the 39th Symposium on the Theory of Computing*, pp. 95–103, New York, NY, USA, 2007. ACM. 50, 55

[Kri01] J. KRINKE. « Identifying similar code with program dependence graphs ». *WCRE*, pp. 301–309, 2001. 31

[Lev66] V.I. LEVENSTEIN. « Binary codes capable of correcting deletions, insertions, and reversals ». *Cybernetics and Control Theory*, 10(8):707–710, 1966. 21

[Lik32] R LIKERT. « A technique for the measurement of attitudes ». *Archives of Psychology*, 1932. 32

[Lip00] C.E LIPSCOMB. « Medical subject headings (MeSH) ». *Bulletin of the Medical Library Association*, 88(3):265, 2000. 52

[LLCF11] C LIM, S LU, A CHEBOTKO, and F FOTOUHI. « Opql: A first opm-level query language for scientific workflow provenance ». In *Services Computing (SCC), 2011 IEEE International Conference on*, pp. 136–143. IEEE, 2011. 45

[LRSS08] W. Lee, L. Raschid, H. Sayyadi, and P. Srinivasan. « Exploiting ontology structure and patterns of annotation to mine significant associations between pairs of controlled vocabulary terms ». In *Data Integration in the Life Sciences*, pp. 44–60. Springer, 2008. 6

[LWMB09] B. Ludäscher, M. Weske, T. McPhillips, and S. Bowers. Scientific workflows: Business as usual? In *Business Process Management*, pp. 31–47. Springer, 2009. 3

[MB00] B.T. Messmer and H. Bunke. « Efficient subgraph isomor- phism detection: a decomposition approach ». *Know- ledge and Data Engineering*, 12(2):307–323, 2000. 32

[MBZL09] T. McPhillips, S. Bowers, D. Zinn, and B. Ludäscher. « Scientific workflow design for mere mortals ». *Future Generation Computer Systems*, 25(5):541–551, 2009. 3

[Mea08] L. Moreau and et al.. « Special Issue: The First Provenance Challenge ». *Concurrency and Computation: Practice and Experience*, 20(5):409–418, 2008. . . . . 8, 16

[MFF$^+$08] L. Moreau, J. Freire, J. Futrelle, R.E McGrath, J. Myers, and P. Paulson. The open provenance model: An overview. In *Provenance and Annotation of Data and Processes*, pp. 323–326. Springer, 2008. 8

[MGRtH11] S Migliorini, M Gambini, M La Rosa, and AH.M ter Hofstede. « Pattern-based evaluation of scientific workflow management systems ». In *Technical report, Queensland University of Technology*, 2011. 44

[Mor06] L. Moreau. « The First Provenance Challenge. », 2006. http://twiki.ipaw.info/bin/view/Challenge/. 12

[MPB10] P. Missier, N. W Paton, and K. Belhajjame. « Fine-grained and efficient lineage querying of collection-based workflow provenance ». In *Proceedings of the 13th International Conference on Extending Database Technology*, pp. 299–310. ACM, 2010. 20

[MSFS11] P. Mates, E. Santos, J. Freire, and C. Silva. « Crowdlabs: Social analysis and visualization for the sciences ». *Proc. of International Conference on Scientific and Statistical Database Management*, pp. 555–564, 2011. 33

[MSRO$^+$10] P Missier, S Soiland-Reyes, S Owen, W Tan, A Nenadic, I Dunlop, A Williams, T Oinn, and C Goble. « Taverna, reloaded ». In M Gertz, T Hey, and B Ludaescher, editors, *Proc. of International Conference on Scientific and Statistical Database Management*, Heidelberg, Germany, 2010. 8

[MvdAW08] A. Medeiros, WM. P. van der Aalst, and A. J. M. M. Weijters. « Quantifying process equivalence based on observed behavior ». *Data Knowl. Eng.*, 64(1):55–74, 2008. 20

[MWHW13] P. Missier, S. Woodman, H. Hiden, and Paul W.. « Provenance and data differencing for workflow reproducibility analysis ». *Concurrency and Computation: Practice and Experience*, 2013. 21

[OAF+03] T.M. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, R.T. Greenwood, K. Carver, M.R. Glover Pocock, A. Wipat, and P. Li. « Taverna: a tool for the composition and enactment of bioinformatics workflows. ». *Bioinformatics*, 20(1):3045–3054, 2003. 20

[PA] PROV-AQ. « http://www.w3.org/TR/prov-aq/ ». 45

[PD] PROV-DM. « http://www.w3.org/TR/prov-dm/ ». 45

[PTK+14] N. Papailiou, D. Tsoumakos, I. Konstantinou, P. Karras, and N. Koziris. « H2RDF+: an efficient data management system for big RDF graphs ». In *ACM SIGMOD international conference on Management of data*, pp. 909–912. ACM, 2014. 20

[RBDL97] T W. Reps, T Ball, M Das, and JR. Larus. « The Use of Program Profiling for Software Maintenance with Applications to the Year 2000 Problem. ». In *ESEC / SIGSOFT FSE*, pp. 432–449, 1997. 21

[RGS09] D.D. Roure, C.A. 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. 3

[RMMTS12] MR. Rodrigues, WC.S. Magalhães, M Machado, and E Tarazona-Santos. « A graph-based approach for designing extensible pipelines ». *BMC Bioinformatics*, 13:163, 2012. 44

[RP10] L. Ramakrishnan and B. Plale. « A Multi-Dimensional Classification Model for Workflow Characteristics ». In *WANDS (ACM SIGMOD)*, 2010. 44

[RWL+06] L. Raschid, Y. Wu, W. Lee, M.-E. Vidal, P. Tsaparas, P. Srinivasan, and A. Sehgal. « Ranking target objects of navigational queries ». In Angela Bonifati and Irini Fundulaki, editors, *WIDM*, pp. 27–34. ACM, 2006. 48

[SAYD+13] J. Stoyanovich, S. Amer-Yahia, S.B. Davidson, M. Jacob, T. Milo, and others. « Understanding Local Structure in Ranked Datasets. ». In *CIDR*, 2013. 48

[SBCBL14] J. Starlinger, B. Brancotte, S. Cohen-Boulakia, and U. Leser. « Similarity Search for Scientific Workflows ». *Proc. of the Int. Conf. on Very Large Data Bases*, 7(12), 2014. . . . . 5, 25, 31, 45, 48

[SCBK+14] J. Starlinger, S. Cohen-Boulakia, S. Khanna, S. Davidson, and U. Leser. « Layer Decomposition: An Effective Structure-based Approach for Scientific Workflow Similarity ». In *Proc. of the 10th IEEE International Conference in eScience*, 2014. 5, 25, 45

[SCBL12] J. Starlinger, S. Cohen-Boulakia, and U. Leser. « (Re)Use in Public Scientific Workflow Repositories ». *Proc. of International Conference on Scientific and Statistical Database Management*, 2012. . . . . 5, 25, 45

[SCM+11] V. Silva, F. Chirigati, K. Maia, E. Ogasawara, D. Oliveira, V. Braganholo, L. Murta, and M. Mattoso. « Similarity-based Workflow Clustering ». *JCIS*, 2(1):23–35, 2011. . . . . 31, 43

[Sea05] DB SEARLS. « Data integration: challenges for drug discovery ». *Nature reviews Drug discovery*, 4(1):45–58, 2005. 2

[SHI] SHIWA. « http://shiwa-repo.cpc.wmin.ac.uk ». 33

[SIY06] P. SHAFER, T. ISGANITIS, and G. YONA. « Hubs of knowledge: using the functional link structure in Biozon to mine for biologically significant entities ». *BMC Bioinformatics*, 7:71, 2006. 48

[SK83] D. SANKOFF and J.B. KRUSKAL. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, 1983. 21

[SKD10] M. SONNTAG, D. KARASTOYANOVA, and E. DEELMAN. « Bridging the gap between business and scientific workflows: humans in the loop of scientific workflows ». In *e-Science (e-Science), 2010 IEEE Sixth International Conference on*, pp. 206–213. IEEE, 2010. 3

[SLA$^+$08] E. SANTOS, L. LINS, J.P. AHRENS, J. FREIRE, and C.T. SILVA. « A First Study on Clustering Collections of Workflow Graphs ». *International Provenance and Annotation Workshop (IPAW)*, pp. 160–173, 2008. 31

[SLDC09] P. SUN, Z. LIU, S.B. DAVIDSON, and Y. CHEN. « Detecting and resolving unsound workflow views for correct provenance analysis ». In *ACM SIGMOD international conference on Management of data*, pp. 549–562, 2009. 20

[Slo07] A. SLOMINSKI. Adapting BPEL to scientific workflows. In *Workflows for e-Science*, pp. 208–226. Springer, 2007. 3

[SM01] J. SESE and S. MORISHITA. « Rank aggregation method for biological databases ». *Genome Informatics Series*, pp. 506–507, 2001. 48

[SNTH13] GK SANDVE, A NEKRUTENKO, J TAYLOR, and E HOVIG. « Ten simple rules for reproducible computational research ». *PLoS computational biology*, 9(10):e1003285, 2013. 4

[SPG08] Y L SIMMHAN, B PLALE, and D GANNON. « Query capabilities of the Karma provenance framework ». *Concurrency and Computation: Practice and Experience*, 20(5):441–451, 2008. 45

[SPSW01] M.Q STEARNS, C. PRICE, K.A. SPACKMAN, and A.Y. WANG. « SNOMED clinical terms: overview of the development process and project status. ». *Proceedings of the AMIA Symposium*, Page 662, 2001. 52

[STD10] J. STOYANOVICH, B. TASKAR, and S. DAVIDSON. « Exploring repositories of scientific workflows ». In *Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science*, Page 7. ACM, 2010. . . . . 31, 43

[SVK$^+$07] C. SCHEIDEGGER, H. VO, D. KOOP, J. FREIRE, and CT. SILVA. « Querying and Creating Visualizations by Analogy ». *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, 2007. 20

[SvZ09] Frans SCHALEKAMP and Anke van ZUYLEN. « Rank Aggregation: Together We're Strong. ». In *ALENEX*, pp. 38–51. SIAM, 2009. . . . . 48, 55

[Tai79] K. TAI. « The Tree-to-Tree Correction Problem ». *J. ACM*, 26(3):422–433, 1979. 21

[TAS09] N TRČKA, WM. AALST, and N SIDOROVA. « Data-Flow Anti-patterns: Discovering Data-Flow Errors in Workflows ». In *Proc. of the 21st International Conference on Advanced Information Systems Engineering*, CAiSE '09, pp. 425–439. Springer-Verlag, 2009. 44

[TKSP12] B TIERNEY, E KISSEL, M SWANY, and E POUYOUL. « Efficient data transfer protocols for big data ». In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pp. 1–9. IEEE, 2012. 23

[TZF10] Wei TAN, Jia ZHANG, and Ian FOSTER. « Network analysis of scientific workflows: a gateway to reuse ». *IEEE Computer*, Page 54, 2010. 43

[Val78] J. VALDES. « *Parsing Flowcharts and Series-Parallel Graphs.* ». PhD thesis, University of Stanford, 1978. 41

[vdAMW06] W.M.P. van der AALST, A. MEDEIROS, and A. J. M. M. WEIJTERS. « Process Equivalence: Comparing Two Process Models Based on Observed Behavior ». In *Business Process Management*, pp. 129–144, 2006. 20

[vdAvHtH⁺11] W. M. P. van der AALST, K. M. van HEE, A. H. M. ter HOFSTEDE, N. SIDOROVA, H. M. W. VERBEEK, M. VOORHOEVE, and M. Thandar WYNN. « Soundness of workflow nets: classification, decidability, and analysis ». *Formal Asp. Comput.*, 23(3):333–363, 2011. 44

[VHR⁺09] R VARADARAJAN, V HRISTIDIS, L RASCHID, M-E VIDAL, L IBÁÑEZ, and H RODRÍGUEZ-DRUMOND. « Flexible and efficient querying and ranking on hyperlinked data sources ». In *Proceedings of the International Conference on Extending Database Technology*, pp. 553–564. ACM, 2009. 6

[VRJ⁺13] K VAHI, M RYNGE, G JUVE, R MAYANI, and E DEELMAN. « Rethinking data management for big data scientific workflows ». In *Big Data, 2013 IEEE International Conference on*, pp. 27–35. IEEE, 2013. 23

[VTL79] J VALDES, R TARJAN, and EL. LAWLER. « The recognition of Series Parallel digraphs ». In *STOC*, pp. 1–12, 1979. . . . . 15, 41

[VTL82] J. VALDES, R. TARJAN, and E.L. LAWLER. « The Recognition of Series Parallel Digraphs ». *SIAM J. Comput.*, 11(2):298–313, 1982. 19

[vZW07] A. van ZUYLEN and D.P. WILLIAMSON. « Deterministic algorithms for rank aggregation and other ranking and clustering problems ». In *Proc. of the 5th Workshop on Approximation and Online Algorithms*, LNCS 4927, pp. 260–273. Springer, 2007. 50, 55

[WF74] R. A. WAGNER and M. J. FISCHER. « The String-to-String Correction Problem ». *J. ACM*, 21(1):168–173, 1974. 21

[Wil94] N. WILDE. « Faster reuse and maintenance using software reconnaissance ». Internal report Technical Report SERC-TR-75F, University of Florida, Software Engineering Research Center, 1994. 21

[WPF05] M Wieczorek, R Prodan, and T Fahringer. « Scheduling of scientific workflows in the ASKALON grid environment ». *ACM SIGMOD international conference on Management of data*, 34(3):56–62, 2005. 3

[WVDVW$^+$09] I. Wassink, P.E. Van Der Vet, K. Wolstencroft, P.BT Neerincx, M. Roos, H. Rauwerda, and TM. Breit. « Analysing scientific workflows: why workflows not only connect web services ». In *Services-I, 2009 World Conference on*, pp. 314–321. IEEE, 2009. 43

[XM07] X. Xiang and G. Madey. « Improving the Reuse of Scientific Workflows and Their By-products ». *ICWS*, pp. 792–799, 2007. . . . . . 31, 32

[YGN09a] U. Yildiz, A. Guabtni, and A. Ngu. « Business versus scientific workflows: A comparative study ». In *Services-I, 2009 World Conference on*, pp. 340–343. IEEE, 2009. 3

[YGN09b] U Yildiz, A Guabtni, and AHH Ngu. « Towards scientific workflow patterns ». In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, Page 13. ACM, 2009. 44

[ZG05] X. Zhang and R. Gupta. « Matching execution histories of program versions. ». In *ESEC/SIGSOFT FSE*, pp. 197–206, 2005. 21

[Zip35] G. Zipf. *The Psycho-Biology of Language.* 1935. 28

[ZS89] K. Zhang and D. Shasha. « Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. ». *SIAM J. Comput.*, 18(6):1245–1262, 1989. 21

[ZTC07] D. Zeginis, Y. Tzitzikas, and V. Christophides. « On the Foundations of Computing Deltas Between RDF Models ». In *Proc. of the International Semantic Web Conference*, vol. 4825 de *Lecture Notes in Computer Science*, pp. 637–651. Springer, 2007. 20

[ZWG$^+$04] J. Zhao, C. Wroe, C. Goble, R. Stevens, D. Quan, and M. Greenwood. « Using Semantic Web Technologies for Representing e-Science Provenance ». In *ISWC*, pp. 92–106, 2004. 20