Mining Frequent Patterns in Bioinformatics Workflows

C. R. Wijesinghe[1,2*], A. R. Weerasinghe[1]

[1] University of Colombo School of Computing, Colombo 7, Sri Lanka.
[2] Faculty of Graduate Studies, University of Colombo, Colombo 7, Sri Lanka.

* Corresponding author. Tel.: (+94) 011-2-581-245; email: crw@ucsc.cmb.ac.lk
Manuscript submitted June 26, 2020; accepted September 13, 2020.
doi: 10.17706/ijbbb.2020.10.4.161-169

Abstract: The goal of workflow systems is to put away the disadvantages of the state-of-the-art methods of scientific data analysis, mostly in Perl or similar scripting languages. Scientific workflow systems enable the development of analysis pipelines, provenance management, process control, recovery, scheduling and parallelization of individual tasks, understandability and sharing of workflows among the scientific community. There are several workflow systems to design bioinformatics workflows. The objective of this work is to identify the frequent workflow patterns or substructures in a corpus of Galaxy bioinformatics workflows obtained from myExperiment. Frequent sub graph discovery (FSG) algorithm used in analyzing the workflows. Seventy-one reusable workflow patterns identified with a 5% minimum support threshold. As future work planning to annotate the identified frequent patterns and to encode the identified patterns in the workflow systems with the objective of improving the usability by providing a high-level abstract interface to the user.

Key words: Frequent subgraph mining, galaxy workflows, substructures in workflows, workflow abstraction.

1. Introduction

Due to the advancement of biotechnology, a massive amount of biological data generated today. Thus, the amount of data processed, analyzed, visualized and managed by scientists enormously increased day by day. For complex analyses, scientists often must combine multiple processing steps into a more massive "analysis pipelines" or "workflows" that can involve several algorithms, specialized computational tools, databases, and web services. Such scientific workflows are executed repeatedly, with different combinations of inputs and parameters. Some of the popular workflow management systems include Galaxy [1], [2], Vistrails [3] and Taverna [4]. Workflows developed using workflow management systems allow users to develop workflows by combining various services and data sources visually. The role of efficient, usable workflow management systems has vital importance in scientific research. The use of workflow systems by the domain experts, for instance, the life scientists in the bioinformatics domain, are limited [5], [6]. Many of the workflow systems can be easily used by computer scientists or trained bioinformaticians but for actual domain experts like life scientists, designing a complex workflow is a challenge. In that sense, we argue that the workflow systems have not reached a level to cater to their actual users yet. The main reason behind this is that many of the bioinformatics workflows are complex and composed of several intertwined tasks making it difficult for general users to design using workflow management systems. Moreover, the reuse of workflows is also tricky since it is difficult to understand the inner processes of complex workflows. With the integration of necessary external services, they become even more challenging to understand.

This study is part of research to make the composition and representation of workflows with workflow systems easy and efficient in order to make them more beneficial to the real users. The specific objective of this study is to identify frequent patterns that can be reused in workflow composition. The claim is that by identifying the patterns or high-level abstractions in workflows, the understandability, reuse, and modularity of the workflows can be enhanced.

1.1. Galaxy Workflow System

Galaxy workflow system developed as a system for the integration of genomic sequences, alignments, and functional annotation of sequences. Galaxy allows users to gather and manipulate data from existing resources in several ways. Recording and storing every action of the user in the history system is a key element of Galaxy. Galaxy enables users to conduct independent queries on genomic data from different sources and to combine or refine them using Galaxy, perform calculations, or extract and visualize corresponding sequences or alignments. Galaxy system provides a new generation of interactive tools for data-intensive genome analysis and allows large-scale analyses that previously required programming experience and database management skills. The history page is identified as simple to use but powerful with the ability to handle large genome annotation data sets. Users can perform multiple types of analyses including query intersections, subtractions, and proximity searches and can display the results using existing browsers like UCSC Genome Browser or Ensembl.

There are 7223 valid tools available in Galaxy ToolShed today. The ToolShed allows Galaxy users to install these freely available Galaxy tools into their local instances. It allows the sharing of tool updates and versions and simplifies the management of the tools. Many labs have their local instances of Galaxy with selected tools categorized into groups. One such categorization is to categorize tools as Text Tools, Genomic File Manipulation, Common Genomic Tools, Genomic Analysis, Metagenomics, Proteomics and Metabolomics, Genomics Tool kits, Statistics and Visualization.

1.2. myExperiment

myExperiment (http://www.myexperiment.org) [7] is an online research environment and the most significant and largest public repository of workflows today. It is a social sharing network since a large number of bioinformatics workflows are available there. These workflows are processes consisting of a series of computational tasks that use web services. Workflows in myExperiment can be accessed by anyone and then be reused and repurposed to their specific requirements.

Moreover, developers submit their workflows to myExperiment and share it with the scientific community. myExperiment encourages users to register to create the social community. By registering as a user in myExperiment, one can get a more affluent user experience than being anonymous. As of today, myExperiment has 10694 members and contains more than 3500 workflows. Similar to other social networks users in myExperiment can request friendship from other registered people having the same research interests. Friendship links can make a network of trusted individuals.

The remainder of the paper is organized as follows. Section 2 describes about workflow abstractions, section 3 presents a comparison of frequent subgraph mining algorithms and FSG, while section 4 summarizes the methodology. Section 5 reports experimental results and discusses the validation of the approach. Later section 6 describes the related work.

2. Workflow Abstractions

Substructures or sub-workflows in scientific workflows represent the usage patterns of the tools. Identifying such substructures help scientists to retrieve, reuse and efficient designing of workflows. Different workflows can share part of their functionality with other workflows.

Several methods have been used in the past in modelling and representing workflows. Graph-based

methods are most popular among those. Graph-based models are based on the theories proposed by [8]. According to [5], definition 1 summarizes the main concepts, just as follows:

A graph G = (V, E) is defined as a non-empty set of vertices (nodes) V = (v1, ..., vn) which are interconnected by an empty or nonempty set of edges (links) E = (e1, ..., em) with E ⊆ V × V and V disjoint from E (V ∩ E = ∅). Graphs are directed when an edge e = (u, v) ∈ E means that the edge goes from u to v (having (u, v) ∈ V), u being the tail of the edge while v being its head. A walk W is a sequence of consecutive vertices (v1, ..., vn) ∈ V such that for each pair of vertices (vi, vi+1) with i ∈ (1, ..., n−1) there exists an edge e ∈ E that connects them. If ∀ (vi, vj) ∈ W vi ≠ vj with i ≠ j, W is denoted as a path. Two vertices of a graph G are connected if there is a path between them. Connection is an equivalence relation on the vertex set V. Hence, a graph G (V,E) can be divided into sets of connected vertices, called components. When G (V,E) has only one component, the graph is connected. Otherwise the graph is disconnected. A cycle C is a walk with (v1, ..., vn), where v1 = vn and where the edges (e1, ..., ek) ∈ C are distinct. A graph without cycles is acyclic. Finally, a graph G = (V, E, LV, LE, ϕV, ϕE) is labeled if LV and LE are sets of labels for the vertices and edges respectively; and ϕV and ϕE are functions that define how each vertex or edge is mapped to a label: V → LV and E → LE respectively.

There are several different types of graphs, including directed graphs, connected graphs, cyclic graphs, and labeled graphs. Scientific workflows are data-intensive and they are often represented as Directed Acyclic Graphs(DAGs). For representing vertices in a graph, an array or a linked list is used. For representing edges adjacency matrix, adjacency list, hash table or trie data structures are used. According to the data structure, memory usage and execution time will be different. Among the different methods, the most straightforward mechanism that can be used to represent a graph structure is by using an adjacency matrix or adjacency list.

3. Frequent Subgraph Mining Algorithms

Workflow mining and frequent subgraph mining are popular domains among the research sub-domains of graph mining. Frequent Subgraph Mining (FSM) is the essence of graph mining, and the objective of FSM is to extract all the frequent subgraphs of a given data set, of which the occurrence counts above a particular threshold.

FSM algorithms work in two steps. First, the candidate subgraphs are in either depth-first or breadth-first manner, and in the next, step the frequency of occurrence of identified subgraphs will be determined. Thus developing an algorithm that will do the above two steps effectively and efficiently is the main focus of FSM research. Further, the FSM algorithms divided into two categories as Inexact FSM and Exact FSM. In inexact FSM algorithms, an approximate measure is used to compare the similarity between two graphs. Hence these algorithms are not guaranteed to find all similar subgraphs, but the computational efficiency is high.

According to the search, strategy, algorithms divided into two categories as Breath First Strategy (BFS) and Depth First Strategy (DFS). BFS guarantees the discovery of all frequent subgraphs and low vulnerability to redundancy. However, DFS is used by most recent algorithms as those consumes less space compared to BFS. The high probability of redundancy in generated subgraphs is high in DFS. [9] has done a comparative survey on graph mining algorithms and has classified the algorithms based on search strategy, nature of input and completeness of output.

A survey done by Jiang *et al.* [10] presents several significant FSM algorithms proposed over the period from 1994 to 2004. According to them, no "new" algorithms were introduced recently but there had been much work on developing variations of existing algorithms. They have also studied the number of FSM algorithms in the literature of different domains and the usage dominated in the chemistry, biology and web domains among others.

They have presented a survey of state-of-the-art algorithms of FSM on many different types of mining strategies concerning different types of graphs producing different types of patterns. Further, the algorithms

categorized based on generation strategy, the mechanism for traversing the search space and the occurrence counting process. According to their survey results, most FSM algorithms adopt exact matching. They have made a comparison of several different exact matching graph isomorphism algorithms employing different techniques and different matching types and have described the limitations and benefits of each algorithm.

We studied 5 different FSM algorithms including Subdue[11], FSG[12], gSpan[13], CloseGraph[14] and Gaston[15] and a comparison of them is shown in Table 1.

Table 1. Comparison of FSM Algorithms

Tool	Input type	Graph representation	Search strategy	Nature of output	Limitation
Subdue	Single graph	Adjacency matrix	Breadth first search	Frequent sub graphs	Small number of patterns
FSG	Set of graphs	Adjacency list	Breadth first search	Frequent connected sub graphs	NP complete
gSpan	Set of graphs	Adjacency list	Depth first search	Frequent sub graphs	Not scalable
CloseGraph	Set of graphs	Adjacency list	Depth first search	Closed connected frequent graphs	inefficient
Gaston	Set of graphs	Hash Table	Depth first search	Maximal frequent sub graphs	Interesting patterns may be lost

3.1. FSG

By considering the pros and cons of each algorithm FSG algorithm was selected to analyze workflows. FSG is a breadth-first search algorithm which provides a complete output with the exact search. A detailed introduction to the FSG algorithm is given in the next section.

FSG[12] algorithm finds all connected subgraphs frequently appear in an extensive graph database. According to the authors, FSG algorithm can achieve excellent performance and scale linearly with the size of the database for problems of moderately large size but for small graphs, the performance of FSG has been worst. A sparse graph representation is used to store input transactions, intermediate candidates and frequent subgraphs. Adjacency list representations are used to store transactions, candidates and detected frequent subgraphs. canonical labeling is used to sort graphs. In finding frequent item sets structure of the Apriori algorithm is used which is more effective in pruning and Transaction ID (TID) has used in frequency counting.

FSG produces three output files as the frequent-pattern file (*.fp) that stores either the frequent or the maximal frequent patterns, parent-children list (PC-list) represents parent-children relationships among the patterns and TID-list file that shows the graph transactions that are supported by each pattern.

3.2. Maximal Pattern Mining

Subgraph mining algorithms need to work with a large number of uninteresting and frequent subgraphs, thus the analysis of frequent subgraphs made impossible. Many applications in graph mining require the result set to be a summary than a complete set of the frequent pattern space. Without enumerating all graph patterns, maximal pattern mining aims to mine a relatively small set of representative patterns with little similarity to each other.

4. Methodology

93 Galaxy bioinformatics workflows were obtained from my Experiment public repository and shared workflows available in the Galaxy platform as .GA files. After removing one-step workflows, example

workflows, and test workflows, a total of 82 workflows were available for analysis. Tools(steps) and connections in the .GA files were obtained using a Python script.

FSG (frequent subgraph discovery) tool was used in extracting the patterns from workflows.

5. Results and Discussion

The analysis was performed with Intel R Core TM i-7 desktop computer with 16GB memory. The input file to FSG contained 82 workflows or 82 transactions. Table 2 shows the summary information about the edges and vertices of the workflows.

Table 2. Summary Statistics of Galaxy Workflows Used in Analysis

Feature	Number
Number of Distinct Edge Labels	417
Number of Distinct Vertex Labels	215
Average Number of Edges in a Transaction	12
Average Number of Vertices in a Transaction	11
Max Number of Edges in a Transaction	86
Max Number of Vertices in a Transaction	68

There were 215 distinct tools were used by the workflows, and among them, tools having a frequency of use of more than 10 were identified as listed in Table 3 with their toolshed repository name. Fastqc which is used to control quality is the most frequently used tool.

Table 3. Tools according to Their Frequency of Use

Tool	Description	Frequency of Use
fastqc	Quality control for high throughput sequence data.	26
cat_multiple	Concatenate_multiple datasets	18
bowtie2	Short read aligner	15
fastq_groomer	Convert between various FASTQ quality formats	13
bowtieForSmallRNA	bowtie wrapper tool to align small RNA sequencing reads	12
fasta_tabular_converter	Fasta file to convert to tabular	12
ncbi_blastn_wrapper	search a nucleotide database using a nucleotide query	12
snpEff	Genetic variant annotation and effect prediction toolbox	12
BlastParser_and_hits	Parses blastn or blastx outputs and organizes hits to subjects	12

The Minimum support threshold used in identifying frequent patterns was 5.0%. The largest pattern size identified was 5. The number of identified frequent patterns with the size of the pattern is shown in Fig. 1. The number of patterns found was decreased with the increase in pattern size.

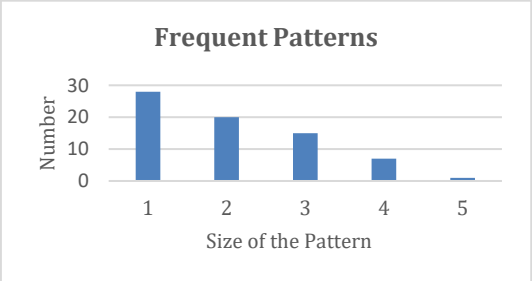

Fig. 1. Frequent patterns

After filtering the repeated and uninterested patterns, 71 patterns were identified. Some of the patterns identified are shown below.

yac&cat_multiple

ncbi_blastn_wrapper&BlastParser_and_hits

oasesoptimiserv&BlastParser_and_hits
bowtieForSmallRNA&oasesoptimiserv
oasesoptimiserv&ncbi_blastn_wrapper
ncbi_blastn_wrapper&BlastParser_and_hits oasesoptimiserv&BlastParser_and_hits
oasesoptimiserv&BlastParser_and_hits oasesoptimiserv&ncbi_blastn_wrapper
oasesoptimiserv&BlastParser_and_hits bowtieForSmallRNA&oasesoptimiserv
ncbi_blastn_wrapper&BlastParser_and_hits oasesoptimiserv&ncbi_blastn_wrapper
ncbi_blastn_wrapper&BlastParser_and_hits oasesoptimiserv&BlastParser_and_hits
oasesoptimiserv&ncbi_blastn_wrapper

In representing the patterns, ampersand is used to show the connection between two tools. Patterns of size one eg. *yac&cat_multiple* can be used to suggest the next tool that is mostly used with a particular tool. By using *yac&cat_*multiple when a user has used *yac* tool suggesting *cat_multiple* as the probable next tool by the system is helpful for a scientist designing a workflow.

Fig. 2 shows the frequent pattern shown below in a diagram. Four tools are connected in this pattern.

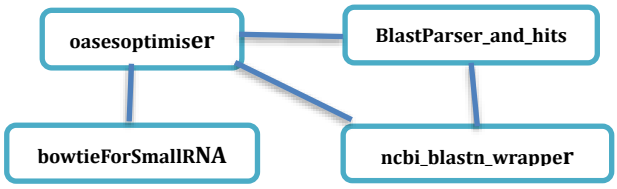

Fig. 2. A substructure identified as a frequent pattern

The processing steps combined in this pattern can be identified as aligning small RNA sequencing reads using *bowtieForSmallRNA,* assemble reads using *oasesoptimiserv,* parses blastn or blastx outputs, and organizes hits to subjects with various coverage information using the tool *BlastParser_and_hits* and search a nucleotide database using a nucleotide query using ncbi_blastn_wrapper.

As described in section 3.2, maximal pattern mining is done to reduce the number of uninterested patterns. Table 4 shows a comparison of frequent patterns found in all frequent pattern search and maximal pattern search.

Table 4. All Frequent Patterns vs Maximal Patterns

	All Frequent patterns	**Maximal patterns only**
Largest Frequent Pattern Size	5	5
Total Number of Frequent Patterns Found:	71	15

The number of patterns has reduced from 71 to 15 when searching only for maximal patterns. Since the total number of frequent patterns found is not considered high, it is not required to consider only the maximal patterns. However, when the number of frequent patterns is very high, it is necessary to consider only the maximal patterns to minimize the number of patterns to consider in the analysis.

Further, we observed the addition of workflows to workflow repositories has reduced over the years. But workflow repositories are of real importance to domain users like life scientists who find difficulties in designing workflows using workflow systems. Searching capabilities within workflow repositories like myExperiment are limited to a keyword search on several tags attached to workflows by authors. However, we observed that the searching within the myExperiment repository with the available tags is not effective. There should be more research on how to make workflow repositories attractive to workflow designers and need to introduce more standards to make repositories more useful.

Finally, when usage patterns or substructures are made available for workflow designers, they will find it easy since it avoids the need for the threading tool by tool. We evaluated the usage of identified patterns with the user community and they found it useful.

6. Related Work

In [16] Garijo *et al.* have proposed an approach to automatically detect motifs from the provenance traces of workflows or set of workflows in a repository. A characteristic feature of their approach is the use of semantic representations to infer generalizations. They used a taxonomy of components associated with a catalog of components in generalization. Further, they demonstrated their method with workflows in the text analytics domain in the Wings workflow system. SUBDUE algorithm applied independently for each of the workflows to retrieve internal macros and in the same way for all workflows to retrieve composite workflows. In the experimental setup, they selected a dataset of 22 workflow templates specified in the Wings workflow system [17] with 30 workflow execution provenance traces of those workflow templates, annotated according to the Open Provenance Model for Workflows (OPMW)[18]. Authors claim that their proposed approach can detect filtered complex fragments successfully while generalizing fragments from workflows and provenance details of their executions.

[19] is an unbiased and comprehensive benchmark of different methods of similarity search in a corpus of workflows. They re-implemented previous methods of similarity search and structured according to a comparison framework. A comprehensive corpus of similarity ratings collected for a set of Taverna and Galaxy workflows and also a gold standard rating obtained from 15 field experts and aggregated using median ranking. All algorithms compared using ranking correctness and retrieval precision.

The study done by [20] proposed a novel principle and an algorithm that can derive all frequent induced subgraphs and association rules is directed or undirected general graphs with loops and labeled nodes and links. Authors also claim that their method can extract topological information about the graphs. A new algorithm developed by extending the Apriori algorithm, mine the graph-structured data with adjacency lists. Codes of the frequent induced subgraphs are derived by a bottom-up approach using canonical form representation of the adjacency matrix. The algorithm implemented using trie data structure and the performance evaluation was done using artificially generated graph-structured transactions.

Diamantini *et al.* [21], have used hierarchical clustering to extract frequent and non-trivial sub-structures and relationships in 258 Taverna workflows. Workflows presented as graphs and SUBDUE which is an inexact matching tool used for clustering and pattern identification. Methodology tested against a subset of myExperiment. Authors claim that their approach can build effective models for representing useful usage patterns and also helps in retrieving workflows that contain specific patterns. Results of the hierarchical clustering model evaluated using several measures.

According to Garijo *et al.*, FragFlow [22] an automatic fragment detection system is developed by integrating additional graph mining algorithms with filters, data preparation and statistical analysis steps to their previous work. FragFlow can be configured to find different sizes or frequencies of fragments, to visualize and also to link them back to original workflows. Fragflow is a combination of three different exact and inexact graph mining techniques i.e. SUBDUE, gSpan and FSG, to find the most common workflow fragments in a corpus of workflows.

In FragFlow, workflows are represented as labeled directed acyclic graphs (LDAGs). The goals of their work were to develop a system that can automatically suggest workflow fragments in a new workflow, find the usefulness of the new fragments that are not suggested by LONI groupings, and study the reusability of workflows. They evaluated their approach by comparing the fragments detected by FragFlow to sub workflows created by the users of the LONI pipeline. LONI pipeline allows users to define groupings in

workflows and it has a library of components with well-defined functionality. A Grouping in the LONI pipeline likely to have an explicit function associated with them explaining their primary role in the workflow.

Conflict of Interest

The authors declare no conflict of interest.

Authors Contributions

C. R Wijesinghe prepared the data set, implemented the method and conducted the evaluation of this study. A. R. Weerasinghe contributed in overall design, identifying conceptual framework, methodology and designing the experiments. All authors contributed in writing the manuscript, reviewed the manuscript critically for scientific content, and all the authors gave final approval of the manuscript for publication.

References

[1] Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., *et al.* (2005). Galaxy: A platform for interactive large-scale genome analysis. *Genome Res., 15(10)*, 1451-1455.

[2] Afgan, E., Baker, D., Batut, B., Beek, M., Bouvier, D., Čech, M., *et al.* (2018). The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Res., 46(W1)*, 537-544.

[3] Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., & Vo, H. T. (2006). VisTrails: Visualization meets data management. *Proceedings of the Acm Sigmod International Conference on Management of Data* (pp. 745-747). Chicago, Illinois.

[4] Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., *et al.* (2004). Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics, 20(17)*, 3045-3054.

[5] Wijesinghe, C. R., & Weerasinghe, A. R. (2019). Experiences on workflow management systems for data-intensive bioinformatics among sri lankan scientists. *Proceedings of Asia Int. Conf. Multidisciplinary Res.* (pp. 63-67).

[6] Deelman, E., Peterka, T., Altintas, I., Carothers, C. D., Kerstin, K. V. D., Moreland, K., *et al.* (2017). The future of scientific workflows. *Artic. Int. J. High Perform. Comput. Appl., 32(1)*, 1-17.

[7] Goble, C. A., Jiten, B., Sergejs, A., Don, C., Danius, M., David, N., *et al.* (2010). Myexperiment: A repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Res., 38(suppl_2)*, W677-W682.

[8] Golosova, O., Henderson, R., Vaskin, Y., Gabrielian, A., Grekhov, G., Nagarajan, V., *et al.* (2014). Unipro UGENE NGS pipelines and components for variant calling, RNA-seq and ChIP-seq data analyses. *PeerJ, 2*, e66.

[9] Krishna, V., Ranga, N. N. R., & Athithan, G. (2011). A comparative survey of algorithms for frequent subgraph discovery. *Curr. Sci., 100(2)*, 190-198.

[10] Jiang, C., Coenen, F., & Zito, M. (2004). A survey of frequent subgraph mining algorithms. *Knowl. Eng. Rev., 00*, 1-24.

[11] Cook, D. J., & Holder, L. B. (1994). Substructure discovery using minimum description length and background knowledge. *J. Artiicial Intell. Res. Submitt., 1*, 231-255.

[12] Michihiro, K., & Karypis, G. (2001). Frequent subgraph discovery. *Proceedings of the IEEE International Conference on Data Mining* (pp. 313-320).

[13] Yan, X., &, Han, J. (2002). gSpan: Graph-based substructure pattern mining. *Proceedings of the IEEE International Conference on Data Mining* (p. 721).

[14] Yan, X., & Han, J. (2003). Closegraph: Mining closed frequent graph patterns. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 286-295).

[15] Nijssen, S., & Kok, J. N. (2004). A quickstart in frequent structure mining can make a difference. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 647-652).

[16] Garijo, D., Corcho, O., & Gil, Y. (2013). Detecting common scientific workflow fragments using templates and execution provenance. *Proceedings of the Seventh International Conference on Knowledge Capture-K-CAP '13* (p. 33).

[17] Gil, Y., Ratnakar, V., Kim, J., Calero, P. G., Groth, P., Moody, J., *et al.* (2011). Wings: Intelligent workflow based design of computational experiments. *IEEE Intelligent System, 26(1)*, 62-72.

[18] Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., & Paulson, P., (2011). The open provenance model. 1-26.

[19] Starlinger, J., Brancotte, B., Cohen-Boulakia, S., Leser, U., & Berlin, Z. (2014). Similarity search for scientific workflows. *Proceedings of the VLDB Endowment* (pp. 1143-1154).

[20] Inokuchi, A., Washio, T., & Motoda ,H. (2003). Complete mining of frequent patterns from graphs mining graph data. *Mach. Learn., 50(3)*, 321-354.

[21] Diamantini, C., Potena, D., & Storti, E. (2011). Mining usage patterns from a repository of scientific workflows. *Proceedings of the Annual ACM Symposium on Applied Computing* (pp. 152-157).

[22] Garijo, D., Corcho, O., Gil, Y., Gutman, B., Dinov, I., Thompson, P., *et al.* (2014). Frag flow automated fragment detection in scientific workflows. *Proceedings of the 2014 IEEE 10th International Conference one-Science* (pp. 281-289).

Rupika Wijesinghe received the BSc. degree in agriculture from the University of Peradeniya, Sri Lanka. She obtained MSc. in computer science and MPhil in computer science from the University of Colombo, Sri Lanka in 2002 and 2012 respectively.

She works as an academic at the University of Colombo School of Computing, Sri Lanka since 2014. Her research interests are in bioinformatics and computational biology. She has made several publications on bioinformatics workflows and cancer genomics.

Ruvan Weerasinghe received his Ph.D. in computer science from the University of Wales College of Cardiff, UK in 1994. He is an academic and researcher at the University of Colombo School of Computing, Sri Lanka.

He has more than a hundred publications in the fields of natural language processing and bioinformatics. His research interests are in most aspects of human language processing, particularly in statistical and corpus-based approaches and also in machine translation. His interests also extend to artificial intelligence, machine learning and bioinformatics.

He has received many international and national awards and grants to carry out research in the area of natural language processing including an ERCIM Visiting Fellowship in 2001 and a fulbright senior scholar award in 2002. He was also visiting professor at the Management & Science University of Malaysia (MSU) in 2010-2011 and visiting faculty at the Umea University of Sweden in 2013.

Dr. Weerasinghe has served as a consultant to numerous projects of national importance in Sri Lanka, mainly via the University of Colombo School of Computing (UCSC) and the Information & Communication Technology Agency of Sri Lanka (ICTA).