Interactive Workflows in a Virtual Laboratory for e-Bioscience: the SigWin-Detector Tool for Gene Expression Analysis

Márcia A. Inda[1], Adam S. Z. Belloum[2], Marco Roos[1],
Dmitry Vasunin[2], Cees de Laat[2], L. O. Hertzberger[3], and Timo M. Breit[1]
(1) Integrative Bioinformatics Unit, (2) System Network and Engineering Group,
and (3) Computer Architecture, and Programming Laboratory Group
Faculty of Science, University of Amsterdam, the Netherlands
{inda, adam, roos, dVasunin, deLaat, bob, breit}@science.uva.nl

Abstract

Explorative research is a vital part of biological sciences. Biologists frequently have to examine and compare multiple (large) sets of biological data in an interactive and explorative manner. Exploring alternative ways of examining the data and managing the necessary resources often require substantial (manual) effort and time.

In this paper, we present a concrete example of how VLAM, a grid-based workflow management system, can enhance experimentation. We discuss in detail the process of developing the SigWin-detector, an application in the domain of bioinformatics. We show that SigWin-detector can promptly identify regions of increased gene expression in transcriptome maps and periodicity in weather data. We also show that the workflow can be extended or partially modified. The individual modules can also be used to compose different experiments. SigWin-detector fulfills the requirements of interactive and explorative experimentation.

1. Introduction

In the course of their research, biologists frequently have to examine and compare multiple (large) sets of biological data. This is the case when, for example, they need to analyze the replicates of a microarray experiment, or compare the data from various tissues of an organism. Ideally, biologists should be able to explore various analysis strategies interactively. Comparing datasets and trying alternative designs and parameter settings whilst continually evaluating results is highly typical of biological research. However, each change of design often requires substantial (manual) effort and the execution of the computations can take a long time. This is a significant bottleneck for a science for which empirical analysis is essential to study its highly complex problems.

A molecular biology researcher needs to manage diverse resources such as genome-sequence related data repositories from web applications (e.g. the UCSC genome browser[1], DNA sequencing equipment, gene expression measurement instruments, and the related data analysis tools. The effective utilization and coordination of such a variety of resources, taken together with the need for support for an explorative and interactive research method, require expertise that is beyond the specialized knowledge of any single scientific field, whether it be biology or any other experimental science. Hence, it is beneficial to adopt an e-Science approach that provides automated support to researchers throughout the whole life cycle of their experiments.

The ideal e-Science framework should deal with the accessibility and management of the required resources, as well as with the creation of a working computational environment containing the necessary analysis tools. Such a an environment is called a *virtual laboratory (VL)*, and can be described as the set of (virtual) facilities with all the necessary infrastructure (both hardware and software) needed to carry out scientific dry-lab experiments and support wet-lab experiments. All this, integrated in a comprehensive and accessible computational environment (see, e.g., [14]).

While the underlying infrastructure for a VL can be provided by grids, the management framework can be provided by *workflow management systems (WMSs)*. Grids supply affordable high performance computing resources and large data storage capability, therefore promising to become the global cyber-infrastructure for the next generation of e-Science applications. Scientific communities utilize grids to share, manage and process large data sets. WMSs can be used to design scientific workflows that automate scientific

[1] `http://genome.ucsc.edu`

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

processes based on data dependencies and their control, as well as to abstract the usage of the required underlying infrastructure to help scientists focus on their own research [1, 12, 2]

In this paper, we discuss how developing workflows supported by a WMS can improve and even open new ways of carrying out interactive and explorative experimentation in a VL. As an example we discuss in detail the process of developing SigWin-detector, an application in the domain of bioinformatics.

2. Workflow Management Systems in e-Science

In order to make resources accessible to the related e-Science communities not only from within the physical location but also through the web, the VL-e program has adopted a (Web) Service Oriented Architecture in combination with WMSs.

Examples of such WMSs for e-Science applications specifically aimed at grids can be found in [10, 8, 7, 19, 2]. The aim of these WMSs is to abstract the details of complexity in the underlying infrastructure from the end user. The users of a WMS need only be concerned with tasks that are essential to achieve their scientific goals, e.g., defining original data sets, defining processes that must be performed and the order in which these processes should be executed.

To obtain insight on how workflow support is provided by the existing systems, a survey of existing WMSs has been conducted. Several well-known systems that we reviewed include: GridNexus [4], ICENI [8], Kepler [1], Pegasus [5], Taverna [12]. The survey focused on three main issues: the experiment design stage, specific requirements for the workflow execution stage (job farming and parameter sweep), and result dissemination. A detailed report, including a series of comparison tables, can be obtained at the gvlam web site[2].

Most of the surveyed systems are under active development and new features are frequently added. All systems evaluated have strong and weak points. For instance: Taverna [12] is a powerful tool to use with Web services and provides direct access to a wide range of bioinformatics services; Kepler [1] is designed to work on standalone computers and has a large library of generic processing components that can speed up the design of the application workflows; and Pegasus [5] has an advanced provenance mechanism. However, the process of creating workflow components in most WMSs is similar. Most systems offer means of wrapping the user's applications into modules that can be controlled and linked to each other by the WMS in question. The complexity of developing new application specific workflow components is almost the same, regardless of which WMS is used.

VLAM offers a good starting point for our application. Firstly, VLAM has native support for developing application components in C and C++, as well as modules written in Java and Python, and it provides a wrapper for legacy binary applications. Secondly, VLAM provides easy access to grid computational resources, and offers a unique feature that allows streaming data between workflow components running on geographically distributed computing resources in an efficient and transparent way. Thirdly, VLAM offers the possibility to modify some of the application parameters at run time without having to restart the execution. This is especially interesting if a calibration phase is needed when some of the application parameters are not known beforehand and need to be tuned.

3. The VLAM Workflow Framework

The VLAM workflow system has been developed in the context of the VLAM-G project by the UvA. The VLAM framework tries to cover the whole life cycle of experimental scientific applications starting from the design phase to the dissemination and sharing of knowledge and of the outcome among a geographically distributed scientific community. The user interface of VLAM is composed of a two-level abstraction workflow, namely: the *Process Flow Template (PFT)* and the *concrete workflow composer* which represents the actual execution of the application on the grid resources [2] (Figure 1).

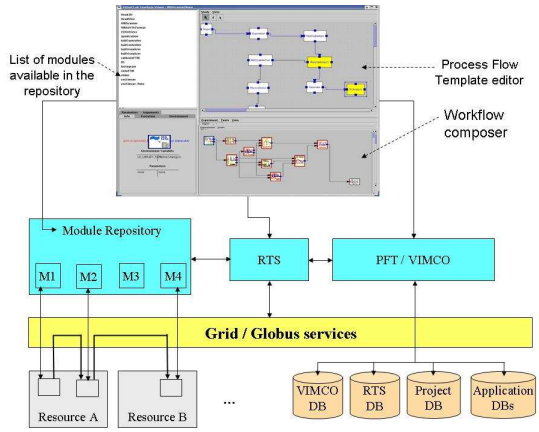

Figure 1. Architecture of the VLAM framework.

Modules form the basic construction elements of the VLAM framework. Users build their experiment connecting the output of a module to the input(s) of other modules.

[2] http://staff.science.uva.nl/~gvlam/doc/P2/SWMSRecommendationReport.pdf

A VLAM module consists of a processing part and a service part. The processing part is the application code, it implements the program logic. The service part provides basic facilities for transferring data between modules and supports the run-time control interface of these modules.

The VLAM *run-time system (RTS)* component is responsible for streaming of the data from an output port of a module to an input port of another module running on a different computing node. The RTS is a data-driven workflow engine designed to provide a transparent and efficient execution of software components on geographically distributed and Grid enabled computational resources. The RTS assumes that all available computing resources have a grid middleware installed (Globus) and have inbound and outbound communications, which allows the creation of data streams between workflow processes located on geographically distributed grid resources.

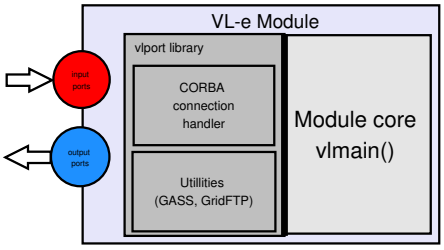

Figure 2. Port library component architecture.

The RTS provides for the application developers a simple API for the creation of I/O ports, a mechanism to interact at run time with the created VLAM modules, and other utility functions (Figure 2). It instantiates, connects, and executes all modules composing the workflow. The input and output ports have a simple interface compatible with standard C++ streams. The VLAM module developers have a simple access to all Grid Access Secondary Storage or GASS data sources using Grid-FTP, FTP, HTTP and HTTPS protocol.

The RTS provides an interface to control predefined parameters of VLAM modules at run-time. This feature allows end-users to interact directly with every process composing the application workflow without having to restart or stop the whole workflow. A parameter in VLAM context is a named entity with a value that can be changed from the modules code as well as from outside, i.e. the VLAM workflow composer (Figure 1). The developer of a module associates a parameter with any arbitrary metric he wants to control. Parameters can be used to monitor the state of a module or to set its state during or before execution. They also can be used to organize a shared blackboard for other subsystems of VLAM.

We are building the SigWin-detector workflow in the VLAM environment to evaluate the capability of a workflow approach, and of the VLAM WMS in particular, to meet the needs of explorative analysis of biological data in a grid environment. In the following section we introduce SigWin-detector. We start by summarizing the required biological background and then briefly describing the computational method.

4. SigWin-Detector: a Bioinformatics Use Case

Loosely speaking, a gene is a section of DNA that contains information on how to produce a certain protein. For this to happen, a gene must first be *transcribed* into its complementary RNA, which is subsequently *translated* into the pertinent protein. The gene in question is said to be *expressed* when translation occurs. Genes are neighboring elements on linear DNA molecules (the chromosomes) and this natural sequential arrangement can play an important role in the regulation of gene expression (see, e.g., [13] and references therein). The importance of chromosomal location is well established in molecular biology, exemplified by the popularity of genome browsers (e.g. [17]) that map many types of information (including genes and their activity or transcription levels) to chromosomal location. Chromosomal location is also used as the basis for a system for integration of distributed genome databases supported by the major providers of biological data [6].

The investigation of profiles of, for instance, gene transcription along the chromosomes (*transcriptome maps*) can identify new phenomena that can improve our understanding of genome function (e.g., [16, 18]). However, in many cases a clear structure is not readily apparent in these profiles. For instance, autocorrelation of directly neighboring genes in the transcription profiles in [18] is very low. Substantial statistical processing is necessary to identify important, but less apparent structures such as *regions of increased (median) gene expression (RIDGEs)* in humans [18], or similar structures in other species (e.g., [3, 15]).

The program originally used to identify RIDGEs in transcriptome maps is called *Ridgeogrammer*[3]. It takes hours to run in a typical desktop computer, lacking the speed and versatility needed for interactive and explorative analysis. Therefore, we developed SigWin-detector workflow, a generalized and improved version of the original program that can be operated interactively in a VL. SigWin-detector takes as input an ordered sequence (a transcriptome map), computes sliding window medians, and identifies as significant windows (RIDGEs) the sliding windows with a median value above a certain *false discovery rate (FDR)* threshold, c.f., [9] (see Figure 3). The term *significant window* is a

[3] http://bussemaker.bio.columbia.edu/software/Ridgeogrammer/

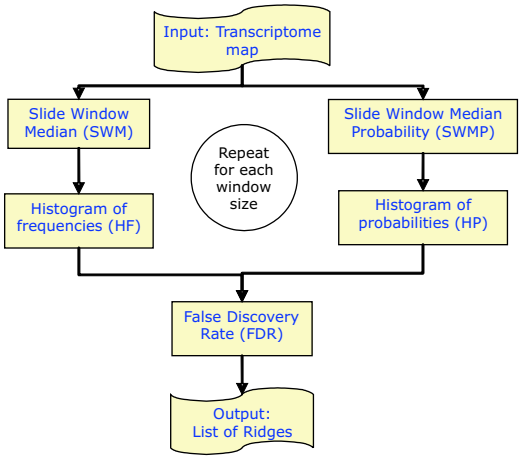

Figure 3. SigWin-detector basic scheme.

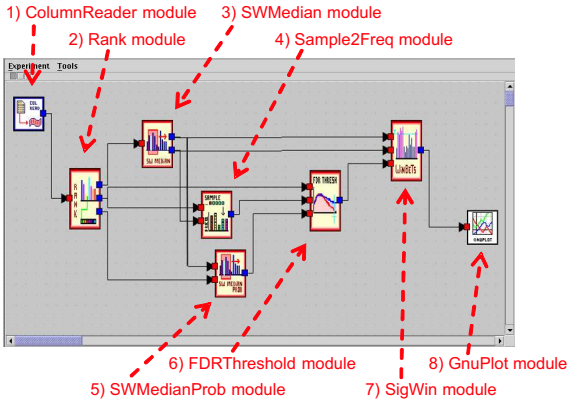

Figure 4. SigWin-Detector basic workflow using the VLAM workflow composer.

generalization of the concept of RIDGE. In this general context, one can analyze different types of genomic profiles or even unrelated sequences such as time series of the temperature.

Our workflow implementation uses a new direct method for computing the null hypothesis distribution needed to compute the FDR thresholds. This direct method is exact (apart from rounding errors) and only needs to compute sliding medians of the input data once, while the Monte Carlo method previously used is approximate and needs to repeatedly compute sliding window medians of permuted input data (approximately 5000 permutations for a sequence of size 25000). This improvement permits a fast identification of the significant windows (minutes instead of hours), opening the doors to an application that can fulfill our expectations of an interactive and explorative analysis tool for genomic sequences.

5. The SigWin-Detector Workflow

The SigWin-detector basic workflow (Figure 4) consists of the following VLAM modules:

1. ColumnReader module: reads the input sequence.
2. Rank module: ranks it.
3. SWMedian module: computes sliding window medians.
4. Sample2Freq module: generates a frequency count from the sliding window median data.
5. SWMedianProb module: generates a theoretical probability density for the sliding window median data from the ranked sequence.
6. FDRThreshold module: applies a false discovery rate (FDR) procedure that compares the obtained frequency count with the theoretical probability to obtain FDR thresholds for each window size.
7. SigWindow module: selects windows above the FDR threshold. And
8. GnuPlot module: displays the results graphically.

Each module performs specific tasks that can be fine-tuned by using the module's parameters. The functionality of each module (including the task to be executed, parameters, and I/O ports) was chosen with the intention of achieving a good balance between functionality and efficiency:

1. Functionality: Each module should do a specific and meaningful task, so that the steps needed to execute the workflow can be easily identified and understood in the larger context of the workflow.
2. Efficiency: The modules should be designed to minimize data communication and task replication by different modules in the same workflow.

To do this, we first determined the various steps needed to complete the desired task (i.e., identify significant windows, see Figure 3), thus emphasizing readability. Then we revised the original scheme taking efficiency issues into account. For example, to compute the sliding window medians it is necessary to compute the rank of each element of the input sequence; this implies sorting it. To compute the theoretical probability density of the sliding window medians it is also necessary to sort the input sequence. Therefore, we added a ranking module to our workflow in order to avoid sorting the same input sequence twice. We also merged steps SWMP and HP of the original scheme (c.f.,

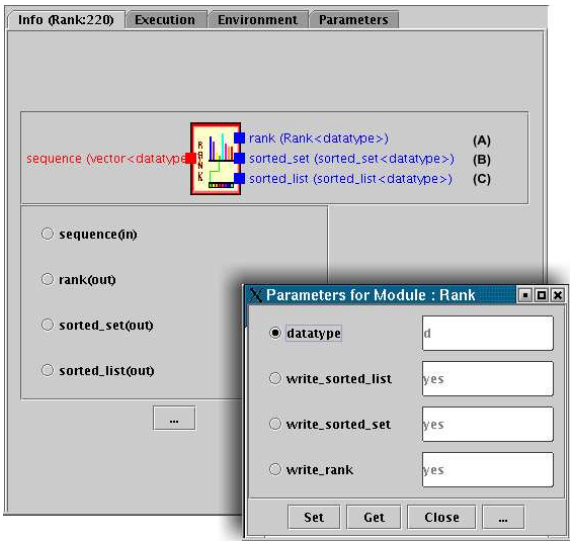

Figure 5. Description of the ports and parameters of the Rank **module.**

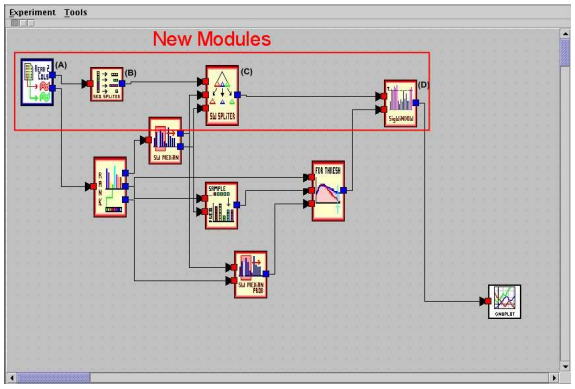

Figure 6. SigWin-detector workflow adapted for displaying significant windows per subsequence. The new modules are: (A) Read2Columns, (B) SeqSplitter, (C) SWSplitter, and (D) SubSigWindow.

Figure 3) into module SWMedianProb of the final workflow, avoiding unnecessary data transfer.

The I/O ports and parameters were also defined taking efficiency reasons into account. For example, the Rank module (Figure 5) has three I/O ports: (A) one for the *rank* structure (a data structure that can access the input sequence in the original order or in the sorted order), required by the SWMedian module; (B) one for the sorted sequence required by the SWMedianProb module; and (C) one for the sorted sequence without duplicates, required by the Sample2Freq and FDRThreshold modules. Each port is associated to a Boolean parameter that controls its output. If any of these parameters is set to false, then the steps necessary to compute the output of the corresponding port will not be performed. This avoids unnecessary computation and communication steps.

The SWMedian module has two output ports, one for the parameters defining the sliding window data structure (sequence size, minimum window size, window size step, and number of window sizes) and one for the sliding window data (the actual values). This is done because some modules need only the parameters, while others need only the data.

Parameters can also be used to fine tune the workflow. For example, if the input sequence is known to be a sequence of integers, one can choose the input data type to be integer, saving both memory space and execution time.

The basic workflow can be altered by substituting, deleting, or adding modules. For example, if we do not want graphical display, the graphics module can be dropped (this is useful for batch processing). Also, we can modify the way the output is displayed by adding (and replacing) modules that split the input DNA sequence and display the RIDGEs (significant windows) per chromosome instead of for the complete DNA sequence. Figure 6 shows the modified workflow, where the ColumnReader module was replaced by the Read2Columns module, which reads two columns of the input file (the first specifying the chromosome, and the second specifying the expression value of each gene). The new modules SeqSplitter and SWSplitter carry out the processes of splitting the sliding window structure into a sequence of sub-structures, each containing data corresponding to a chromosome. The old SigWindow module is replaced by the new SubSigWindow module that can handle multiple sliding window structures.

Converting an existing application into a VLAM module is straightforward. It requires minor changes to the original code and no knowledge of grid middleware is needed. For C++ applications, a module is represented by a C++ class that inherits from VL::VLApplication. The VL::VLApplication class provides methods to create an input or an output port. It also provides a method for getting the value of a parameter from the user interface. For details we refer the reader to the VLAM user guide[4]. It is also possible to wrap a stand alone application using the legacy application wrapper (*lawrapper*). In this case, we only need to provide a configuration file containing the information lawrapper needs: the number and the type of input/output ports; and the command that have to be executed. The module GnuPlot was wrapped this way.

[4] http://staff.science.uva.nl/~gvlam/pub/gvlam-dist/stable-1.4/UserGuide.pdf

6. Using the Workflow

Once a workflow is composed, we can run it or save it for later use. We tested our basic workflow and some alternative ones in a grid cluster composed of geographically distributed computational nodes. For this experiment we used the Distributed ASCI Supercomputer 2 (DAS-2) (www.cs.vu.nl/das2). Figure 7 shows a typical snapshot of an interactive section. The figure shows a graphical output of the RIDGEs (significant windows) encountered in the human transcriptome map compiled by Versteeg and coworkers [18]. Each triangular graph (*ridgeogram*) corresponds to a chromosome. Each row in a graph represents a window size (ranging from 1 to N, the size of the input sequence). Each column represents a sliding window number (ranging from $w/2$ to $n - w/2$, where w is the window size), hence the triangular form. Each RIDGE is identified by a point in the graph, and the colors represent the actual sliding window median expression value. Our results agree with the results of Versteeg et al. [18].

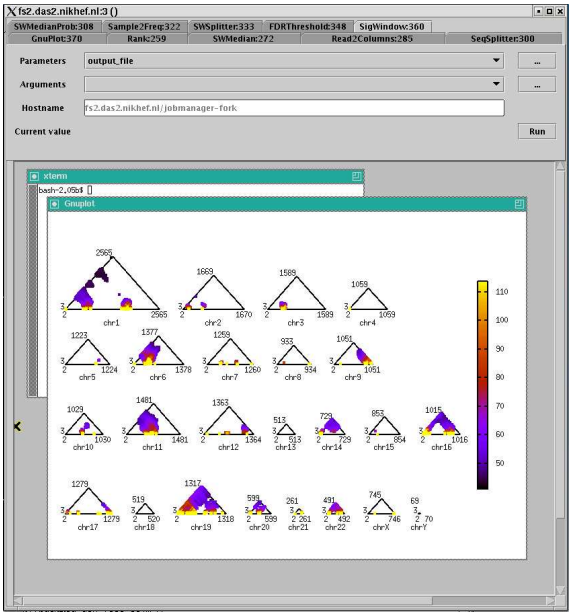

Figure 7. Graphical output of the modified SigWin-detector workflow taking as input the human transcriptome map compiled in [18]. The VLAM WMS distributes the workflow on grid-enabled resources and redirects any graphical output generated to the default user screen.

Table 6 presents some preliminary performance results. In Table 6a the step size is kept constant and in Table 6b it grows with N.

Table 1. Timing results for running SigWin-detector basic workflow.

a) step size = 2		b) step size = $N/1000$	
N	time (s)	N	time (s)
100	21	2000	59
500	22	20000	115
1000	34	40000	420
2000	59	80000	621
5000	192	100000	824
10000	425	120000	930

Being able to choose the parameters for the sliding window structure can be useful when doing explorative data analysis. In the beginning one could set the maximum window size range with a large step size to get an overview of the distribution of significant windows. At a later stage a more detailed picture of the interesting parts can be obtained by reducing the window size range and choosing a small step size.

SigWin-detector has a wider range of application. Figure 8 shows SigWin-detector being applied to a time series of temperature in Amsterdam. In this case a significant window corresponds to a warmer than median period. Note that if the window size is larger than one year, the significant window pattern is reversed, forming the checkerboard pattern characteristic of periodical sequences. This happens because, if the window size is larger than the period (a year), a window centered in the winter will have more warm than cold days and be marked as significant. Conversely, a window centered in the summer will have more cold than warm days and will not be marked.

7. Conclusions And Future Work

Biologists and the experimental scientific community as a whole are in great need of methods and tools to enable explorative and interactive research. They also need ways to coordinate the resources they utilize effectively. Virtual Laboratories (VL) have emerged as part of an e-Science approach aimed at solving these problems.

A VL should provide the necessary infrastructure needed to carry out scientific experiments, as well as a working computational environment containing the necessary analysis tools. While the underlying infrastructure for a VL is provided by grids, the management framework is provided by workflow management systems (WMSs). In this paper, we presented a concrete example of how WMSs can create an environment where biologists can perform their (computational) experiments on the grid, without having to deal with the complexities of the underling grid middle-

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

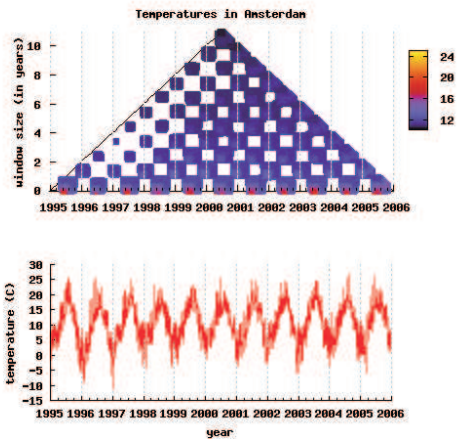

Figure 8. Temperatures in Amsterdam. The significant window regions correspond to the warmer months of the year.

ware, thus allowing them to focus only on the biological aspects of their experiments.

We have developed SigWin-detector, a publicly available interactive workflow that identifies significant windows in a given sequence of values. It can be used to analyze transcriptome maps and other types of genomic profile. In this way, we used SigWin-detector to identify regions of increased gene expression in the human transcriptome map compiled by Versteeg and coworkers [18]. Our results agree with their results. However, SigWin-detector has a wider range of application and it can be used with any ordered sequence. As an example, we tested our workflow using a time series of temperature in Amsterdam. The resulting checkerboard pattern of significant windows clearly revealed the periodicity of the data, identifying the periods of temperature higher than the median (i.e., summers).

The SigWin-detector workflow was implemented using the VLAM workflow management system. This implementation takes advantage of some of the features of the VLAM workflow namely: (a) the interactive creation and execution of workflows in a grid environment; (b) the automatic redirection of the graphical output to the end-user default screen; (c) the possibility to adapt the workflow to meet the user's specific needs; (d) the ability to run an experiment in batch mode by calling directly the run time system from a script.

Furthermore, the modular design discloses the structure of an experiment to biologists in a more testable way than through written publications and poorly reproducible scripts. It facilitates optimization of parts of the workflow and enables experimentation with variations of the original design. Also, the workflow itself can be used as a 'module' in more elaborate experiments.

At the moment we are running our workflow using different kinds of input data, including DNA sequence profiles, microarray data, and weather-related data. We also intend to run it using different types of fabricated data to create a collection of patterns so that we can identify such signatures when they appear in an experimental input sequence. We are also investigating how additional background knowledge about the data can be automatically incorporated in the system [11]. This information will be needed for the interpretation of increasingly complex experiments made possible by VLs.

The VLAM WMS is currently being extended with new features which will improve both the composition and execution of the application workflows. The VLAM RTS will be implemented as a WSRF-compliant web service allowing end-users to connect from various locations to follow the execution of their workflow. Another new feature is to replace the current centralized information systems to keep track of the VLAM module's descriptions and characteristics. In the new version a discovery system based on peer-to-peer network will be added. This new component will provide up-to-date information about available VLAM modules and resources and can be used by the resource manager in the optimization of the schedules. Another important feature we are currently developing is the interoperability with other WMSs, the objective being to allow a user to develop his application workflow using components which have been developed in different WMSs. A meta-workflow engine, which we call *WorkflowBus*, will coordinate the execution of the used engines.

Acknowledgments We would like to thank dr. Marcel van Batenburg for fruitful discussions and the bioinformatics group of dr. Antoine van Kampen for making the human transcriptome map data available to us.

This work was carried out in the context of the Virtual Laboratory for e-Science project (www.vl-e.nl). Part of this project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

This work was part of the BioRange program of the Netherlands Bioinformatics Centre (NBIC), which is supported by a BSIK grant through the Netherlands Genomics Initiative (NGI).

References

[1] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *SSDBM*, pages 423–424, 2004.

[2] A. Belloum, D. Groep, L. Hertzberger, V. Korkhov, C. T. de Laat, and D. Vasunin. Vlam-g: A grid-based virtual laboratory. *Future Generation Computer System*, 19(2):209–217, 2003.

[3] A. M. Boutanaev, A. I. Kalmykova, Y. Y. Shevelyov, and D. I. Nurminsky. Large clusters of co-expressed genes in the Drosophila genome. *Nature*, 420(6916):666–669, 2002.

[4] J. L. Brown, C. S. Ferner, T. C. Hudson, A. E. Stapleton, R. J. Vetter, T. Carland, A. Martin, J. Martin, A. Rawls, W. J. Shipman, and M. Wood. Gridnexus: A grid services scientific workflow system. *The International Journal of Computer and Information Science (IJCIS)*, 6(2), 2005.

[5] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus: Mapping scientific workflows onto the grid. In *European Across Grids Conference*, pages 11–20, 2004.

[6] R. D. Dowell, R. M. Jokerst, A. Day, S. R. Eddy, and L. Stein. The distributed annotation system. *BMC Bioinformatics*, 2:7, 2001.

[7] R. Duan, T. Fahringer, R. Prodan, J. Qin, A. Villazon, and M. Wieczorek. Real world workflow applications in the askalon grid environment. In *European Grid Conference (EGC 2005)*, Lecture Notes in Computer Science. Springer Verlag, February 2005.

[8] N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington. Iceni: An open grid service architecture implemented with Jini. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–10, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[9] Y. Hochberg and Y. Benjamini. More powerful procedures for multiple significance testing. *Stat. Med.*, 9(7):811–8, 1990.

[10] S. Majithia, M. S. Shields, I. J. Taylor, and I. Wang. Triana: A Graphical Web Service Composition and Execution Toolkit. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 514–524. IEEE Computer Society, 2004.

[11] M. S. Marshall, L. Post, M. Roos, and T. M. Breit. Using semantic web tools to integrate experimental measurement data on our own terms. In *submitted to the International Workshop on Knowledge Systems in Bioinformatics (KSinBIT'06)*, 2006.

[12] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[13] B. Oliver, M. Parisi, and D. Clark. Gene expression neighborhoods. *J Biol*, 1(1):4, 2002.

[14] H. Rauwerda, M. Roos, L. Hertzberger, and T. Breit. The promise of a virtual lab in drug discovery. *Drug Discovery Today*, 11(5-6):228–236, 2006.

[15] P. J. Roy, J. M. Stuart, J. Lund, and S. K. Kim. Chromosomal clustering of muscle-expressed genes in Caenorhabditis Elegans. *Nature*, 418(6901):975–979, 2002.

[16] P. T. Spellman and G. M. Rubin. Evidence for large domains of similarly expressed genes in the Drosophila genome. *J. Biol.*, 1(1):5, 2002.

[17] UCSC. Ucsc genome browser.

[18] R. Versteeg, B. D. van Schaik, M. F. van Batenburg, M. Roos, R. Monajemi, H. Caron, H. J. Bussemaker, and A. H. van Kampen. The human transcriptome map reveals extremes in gene density, intron length, GC content, and repeat pattern for domains of highly and weakly expressed genes. *Genome Research*, 13(9):1998–2004, 2003.

[19] F. Xu, M. H. Eres, D. J. Baker, and S. J. Cox. Tools and support for deploying applications on the grid. In *IEEE SCC*, pages 281–287, 2004.

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE