J Grid Computing (2020) 18:507–527
https://doi.org/10.1007/s10723-020-09518-y

Design of a Flexible, User Friendly Feature Matrix Generation System and its Application on Biomedical Datasets

M. Ghorbani · S. Swift · S. J. E. Taylor · A. M. Payne

Received: 26 June 2018 / Accepted: 23 March 2020 / Published online: 27 April 2020

Abstract The generation of a feature matrix is the first step in conducting machine learning analyses on complex data sets such as those containing DNA, RNA or protein sequences. These matrices contain information for each object which have to be identified using complex algorithms to interrogate the data. They are normally generated by combining the results of running such algorithms across various datasets from different and distributed data sources. Thus for non-computing experts the generation of such matrices prove a barrier to employing machine learning techniques. Further since datasets are becoming larger this barrier is augmented by the limitations of the single personal computer most often used by investigators to carry out such analyses. Here we propose a user friendly system to generate feature matrices in a way that is flexible, scalable and extendable. Additionally by making use of The Berkeley Open Infrastructure for Network Computing (BOINC) software, the process can be speeded up using distributed volunteer computing possible in most institutions. The system makes use of a combination of the Grid and Cloud User Support Environment (gUSE), combined with the Web Services Parallel Grid Runtime and Developer Environment Portal (WS-PGRADE) to create workflow-based science gateways that allow users to submit work to the distributed computing. This report demonstrates the use of our proposed WS-PGRADE/gUSE BOINC system to identify features to populate matrices from very large DNA sequence data repositories, however we propose that this system could be used to analyse a wide variety of feature sets including image, numerical and text data.

Keywords BOINC · Desktop grid · DNA sequence · Feature subset selection · Machine learning · High performance computing · WS-PGRADE · gUSE · DNA feature identification · DNA sequence · Speedup

M. Ghorbani · S. Swift · S. J. E. Taylor · A. M. Payne (✉)
Department of Computer Science, College of Engineering, Design and Physical Sciences, Brunel University London, Kingston Lane, Uxbridge, Middx UB8 3PH, UK
e-mail: annette.payne@brunel.ac.uk

1 INTRODUCTION

Machine learning techniques have proved to be important tools in many research areas to aid knowledge discovery from complex data sets. Examples of its far reaching impact and methods have been extensively reported [1–3]. Machine learning analysis however is preceded by the important stage of feature matrix generation which selects the features to be analyzed from these data sets. In some cases these features can be simply a chosen subset of features in the data set; chosen using expert knowledge of the subject arena the data was collected from. Often however the features are generated by running algorithms across the data to draw out derived features or values not in the original data set. It follows therefore that the successful outcome of machine learning techniques is highly dependent upon the feature generation stage [4]. This adds an additional layer of complexity to an already difficult analysis for the non-expert.

Table 1 methods for grid enabling in BOINC

	BOINC API	DC-API	BOINC Wrapper	GenWrapper	GBAC
Supported Programming languages	C/C++/FORTRAN/Python	C/C++/Java/Python	Control-flow description in XML	POSIX shell scripting	non required
Legacy application	No	No	Yes	Yes	Yes
Native application	Yes	Yes	Partial	Partial	Partial
Application level checkpointing	Yes	Yes	Partial	Partial	Partial
Type	Native	Native	Native	Native	Virtualized
Requires client-side third party software	No	No	No	No	Yes (VirtualBox predeployed)

A significant amount of work has been achieved [5, 6] focusing on the algorithms for feature generation ensuring that the features in the matrix are of high quality, however we can find no work describing a system designed to place these algorithms into a kind of workflow that makes this stage flexible. Thus we noted a gap in systems design that allowed interchange of feature generation algorithms so if new or different features needed to add to the matrix this can be done with little impact on the original scheme. Currently feature generation is prepared de novo from first principles for each particular analysis in a project and there is no general propose flexible design for this stage of machine learning. We concluded that one of the best ways to do the feature generation is the use of workflow systems [7]. This enables the workflow design to be later reused or merged with other workflows. Also if there is a need to focus on a particular class of features, the workflow system could be enabled to turn off or on that part of the workflow, and see the resultant outcome. Current work flow systems include Nextflow https://www.nextflow.io/, Snakemake, [8] Galaxy [9] however these systems are either complex to use, not as flexible as we required or if they are easy to use have not been implemented on a distributed system important for the analysis of large datasets. To test our system we decided to use some of the most complex data sets currently available namely those in the field of biology and biomedicine, with particular reference to molecular genetics. In choosing this field we were also driven to provide not only a flexible system but a very user friendly one since most experts in this filed have little computational expertise.

Today's studies in biology and biomedicine involve ever increasing complex and larger data sets which the investigators wish to data mine for new discoveries and novel associations. It has been described as the fourth paradigm of data-intensive science [10, 11] or data-driven science after the first three paradigms of computational, experimental and theoretical science. This data-driven science complements existing paradigms attempting to link knowledge with observations [12]. This new discipline requires new data management systems which are distributed in many locations. Distributed and cloud computing capacity are needed to cope with the enormity of the data [11]. Further the data may be in different formats and the experimenter may wish to make use of many very different software packages to achieve the analysis required. Cross-architectural implementation of algorithms and systems is being attempted to analyse this data (e.g., Grid/Cloud [13–22]) but these efforts are only achievable with the help of computer specialists e.g. the HUBzero [17] cyber grid infrastructure or they are limited to workflows within a particular topic area e.g. WeNMR for structural biology with access to a computing grid offered by the project partners [16]. This poses issues for the data domain experts as they try to grapple with unuser-friendly software and the limited computer processing capacity of their single computers if they choose to undertake bespoke analyses. The workflows we have developed to assist in these analyses try and map "routes" through the steps need to analyze a particular data set in the way desired by the experimenter and assist in creating bespoke successions of processes that need to be carried out on a data set to analyze it. Further they allow users to use volunteered distributed systems to harness greater processing power and to achieve practical run times [19, 20].

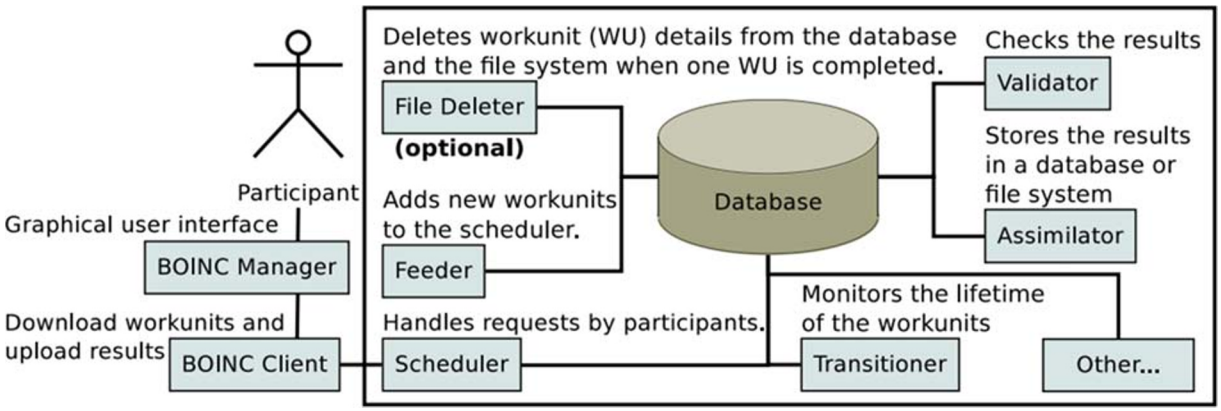


Fig. 1 Interaction between different BOINC processes, database processes and the server side components (shown in the rectangles) [29].

Many bioinformatics software packages and algorithms exist for the identification of biological features. Some of them are available as user friendly web-based software with a GUI interface such as MEME which identifies DNA motifs (meme-suite.org/) or those on the NCBI hub (https://www.ncbi.nlm.nih.gov/). However, the vast majority are only available as command line tools and so are inaccessible to the non-computationally trained biologist. This work details the design of a system that uses volunteer distributed computing to query and explore biological data using a user-friendly and form-based workflow as a way to make these packages and algorithms more accessible for the non-computer coding expert. The user can easily interact with the application like any other web-based application without the need to know the details of the workflow, making it easy to use by non-programmers. The gUSE workflow portal was selected as the workflow management service as it can be connected to a diverse range of distributed computing infrastructures. The advantage of using gUSE are an application specific API (Application Program Interface) which enables the developer to develop a form-based interface for workflows therefore the user can easily interact with the application without the need to know the details of the workflow. WS-PGRADE/gUSE [18] was chosen as it offers a complete, customizable web-based generic portal framework system to run applications. Different applications, each representing a function to be executed on the data, can be joined to create customized

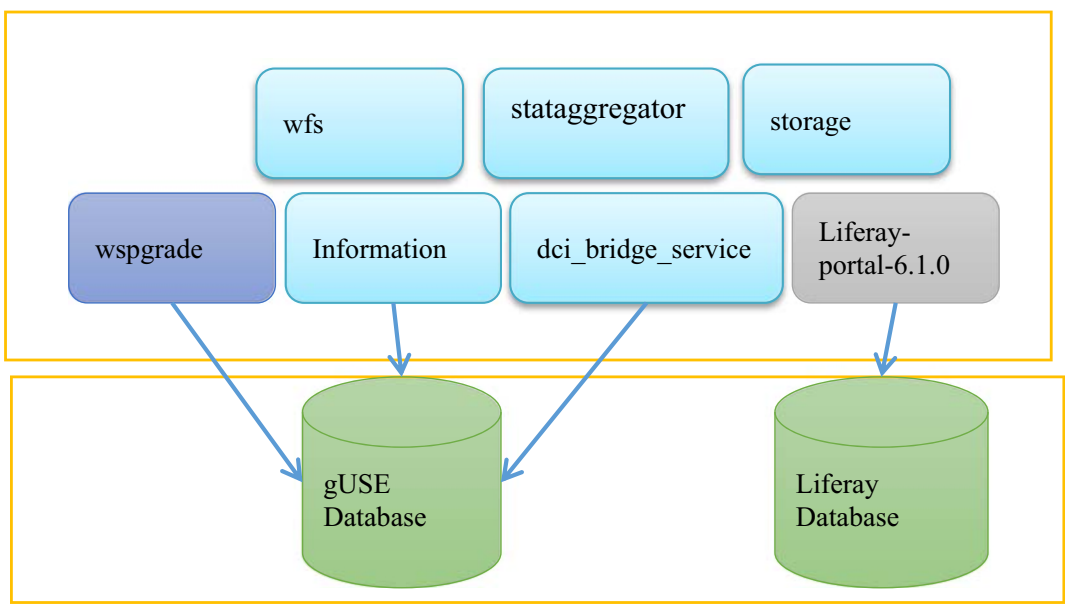


Fig. 2. gUse server architecture.

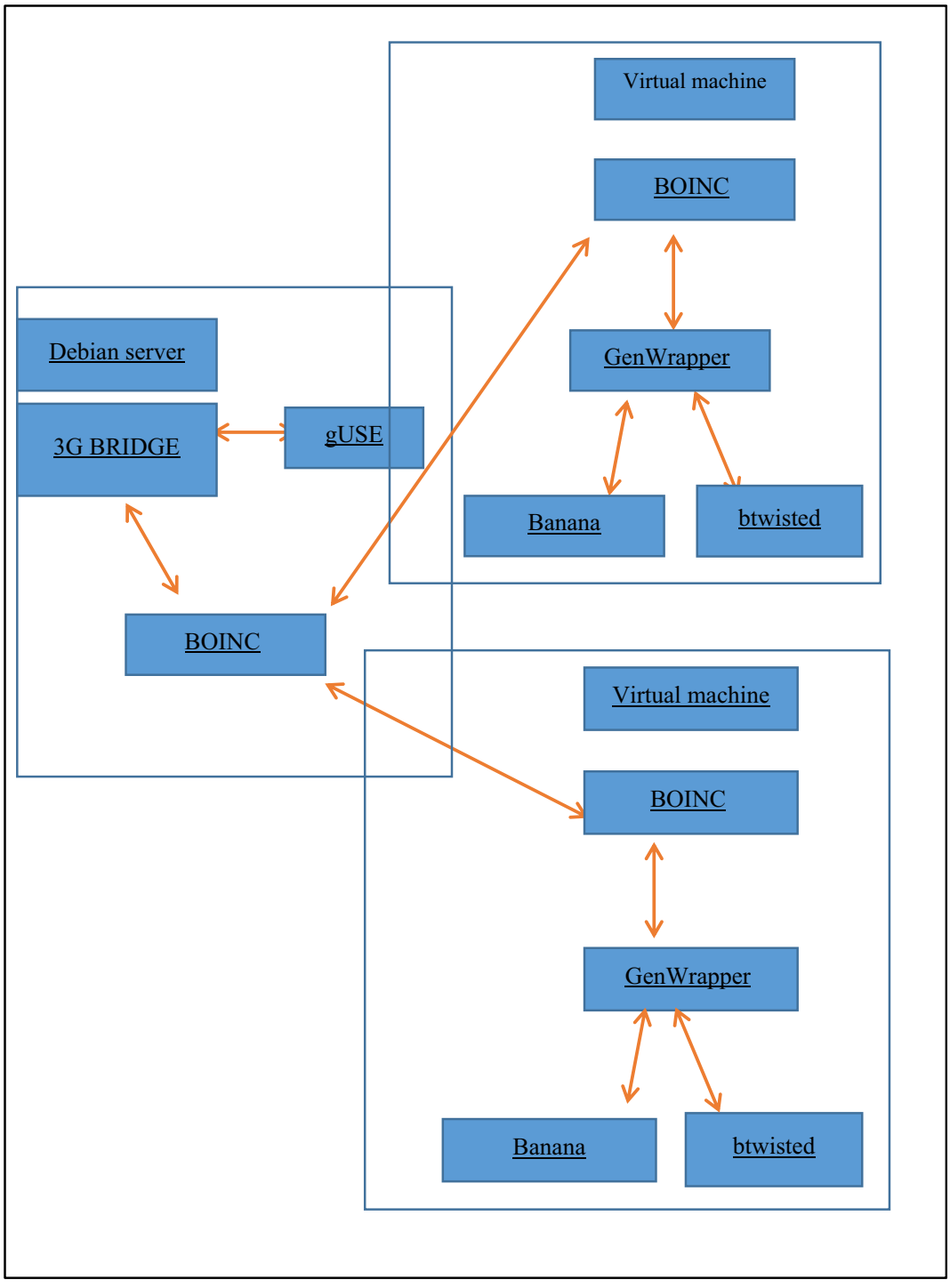


Fig. 3 The different software used in the client and server side of the system showing their interaction with each other

workflows. Details of the WS-PGRADE/gUSE workflow concept is described in Balasko (2014) [23].

In our experience many biologists do not have access to clusters or dedicated High Performance Computing facilities which offer the ability to speed up the extended

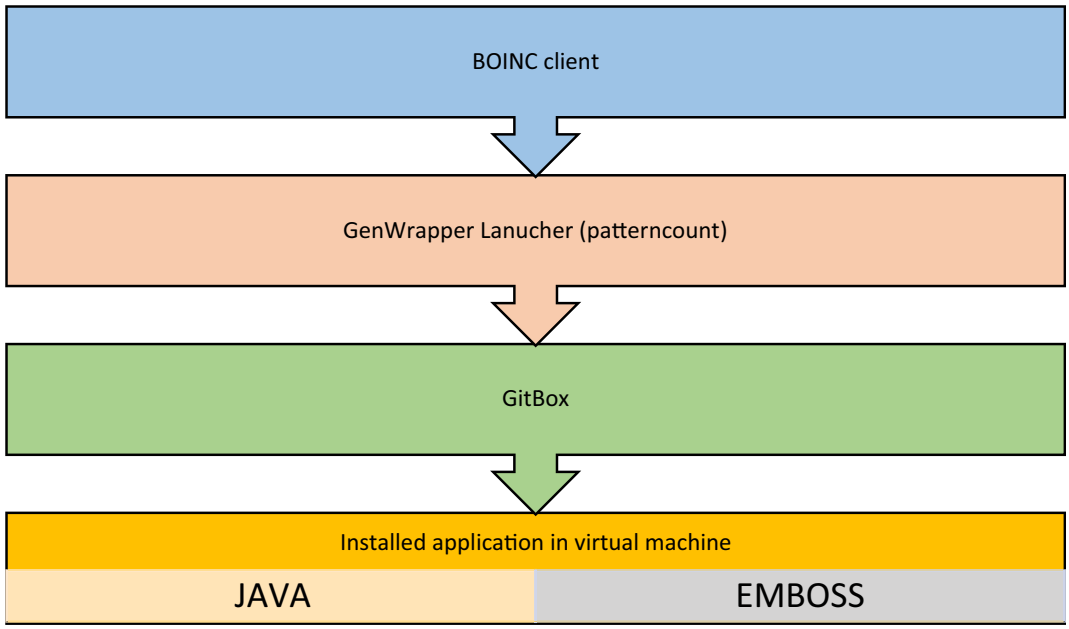


Fig. 4 The client side view of program used for feature generation and feature subset selection

complex processes they wish to run. We therefore wanted to extend our workflow approach by using commonly available commodity PCs or desktop grids to offer not just grid computing capability but volunteer grid computing. The workflow management service provides a modular way of feature generation, then by connecting the workflow to a volunteered distributed system it provides the facility to speedup feature generation and feature subset selection tasks. There have been many advances in recent years in the organized provision of computing resources for computationally demanding applications. Cloud computing, Grid computing and e-Infrastructures, for example, bring together multiple computing resources over multiple domains, or distributed computing infrastructures [24]. One area of interest within this is the use of volunteer grid computing and the multiple desktop computers of an enterprise. There are several middlewares available for desktop grid computing. These include e.g. gLite [http://glite.cern.ch], ARC [25], HTCondor [26], Globus [27] or and BOINC [28]. However, many of these require advanced computing skills and are difficult for non-expert users to use. A way around this issue is to make use of science gateways [29–33]. These are user-friendly, easy-to-use web interfaces that enable end-user scientists to run their experiments quickly and without the need to learn the particular features of the distributed infrastructure. WS-PGRADE/gUSE is an example of one of these gateways which offers a complete workflow-based framework which has flexible customization methods and enables job submission to a variety of distributed computing infrastructures. Bioinformatics applications have applied a volunteer desktop grid approach for solving computation intensive tasks previously: The project Folding@Home simulates protein folding and molecular dynamics. Folding@Home claimed that it is the largest distributed computing projects with over 8,300,000 CPUs participating in the project [34]. Part of the software is proprietary and part is open-source. Similar to this project is the Rosseta@Home project which uses the BOINC platform to submit jobs and download the results (https://boinc.berkeley.edu/). BOINC has also been shown to be effective with the bioscience application GPUGrid.net [34–36]. This work therefore investigates how BOINC could be leveraged with WS-PGRADE/gUSE so that jobs could be submitted to a volunteer desktop grid to offer that processing speed up required to make analyses not only possible but additionally completed in realistic time frames. BOINC has proved to be a useful resource for analyses that require a relatively fast turnaround time so we sort to utilise it in this study [35].

The contribution of this work is the creation of a novel gateway system that is truly non-expert user friendly to generate feature matrices in a way that

cpg_id	banana_bend_structure	banana_curve_structure	btwisted_turn_structure	btwisted_twist_structure	btwisted_stack energy_structure	ATCG	TTGC	TTGG	TCTA	TTGA
>ExprA_GRCh37_11__47416487_cg10511890_199680_CpG	17.4049	6.76311	4140.8	11.5	-1038.28	0	1	0	0	0
>ExprA_GRCh37_11__60739005_cg27284288_475867_CpG	17.4615	3.16885	4102.4	11.4	-1025.16	0	0	1	0	0
>ExprA_GRCh37_11__60739172_cg07380416_144123_CpG	15.7279	4.12377	4096.1	11.38	-1003.81	1	0	1	0	1

Fig. 5 Input FASTA sequence file and Output feature file output of the banana program which is a profile file and has a value for each base in the sequence

is flexible, scalable and extendable. Additionally by making use of BOINC software, to propose that the processes can be easily and affordably speeded up to allow execution in realistic timeframes by utilising a volunteer distributed computing network, available in most institutions.

In order to demonstrate the system it was trialed by undertaking the task of feature identification from repositories of DNA sequences. The system generates features or can query the sequence around existing features in DNA sequences. We demonstrate the system is scalable and editable, so if new features need to be investigated, the whole system does not need to be altered. Similarly, we show how the system could equally be used to analyze RNA or protein sequences.

With the advent of the genome sequencing projects and the ENCODE project the amount of sequence data has grown exponentially. A large amount of this data is stored in public and private databases and most is freely available. For example, the European Bioinformatics Institute (EBI) stores 75 petabytes of data and its databases are being added to on a daily basis [37]. One challenge is now to determine novel features and their roles, investigate known features, determine what the significance is of these features, how they influence gene expression and how they interact with each other. Understanding these features is crucial to the understanding of the biology of living things in both health and disease. Thus mass analysis of sequence data is now crucial to identify trends, motifs, commonality and differences between features, individuals and species. Novel features need to be identified de novo by their proximity to known features or their association with biological phenomenon. Known features are studied for associations with phenomenon or other known features to better understand their significance and role. Well known examples of known features are transcription factor binding site consensus sequences. Phenomenon that such features can affect include methylation, DNA condensation and modifications which ultimately lead to differences in gene expression and thus phenotypic differences.

Many features can influence DNA, RNA and protein structure, modification or gene expression and many new features are being discovered as technology and science progresses. Identifying novel or known features within

Algorithm 5. Sequential feature generation algorithm	
Input:	DNA sequences
1.	For each DNA sequence
2.	For each feature generation program
3.	Run the feature generation program on the Sequence
4.	Store the result in the feature matrix
5.	End For
6.	End For
Output:	Feature matrix

Fig. 6 Sequential feature generation algorithm

Base	Bend	Curve
C	0.0	0.0
A	19.5	0.0
C	17.7	0.0
C	17.3	0.0
T	11.7	0.0
T	6.8	0.0
A	9.0	0.0
G	14.3	0.0
Continued ….		

Fig. 7. banana profile file.

sequences de novo remains a challenge due to the large amount of sequence data now deposited in the databases. As a research project case study for the system we used it to identify features in the DNA sequence surrounding the thousands of CpG sites in genomic DNA. CpG sites are regions of DNA where a cytosine nucleotide is followed by a guanine nucleotide in the linear sequence of bases along its 5′ → 3′ direction. They are of great interest to biologists as they are implicated in gene expression and many disease mechanisms. This study was ideal because generating features for each CpG site is independent of other CpG sites, these tasks can be distributed over multiple machines.

The main purpose of feature generation is to define characteristics of data based on some measurement. So, if there are *n* different data points and m different features for each data item, tabular data matrix of size $n \times m$ can be generated. This table then will be used in data mining methods. This data matrix can be later used to find out how well all features can classify objects or which feature subset better classifies data items. Sometimes features can be categorized into different groups based on defined criteria. These groups can be independently used in the classifier. Classification performance for each group can indicate the importance of each feature group in the study. In the case of CpG sites for example, each CpG site represents one data item and the sequence around the CpG site is used for generating features. This approach has been used for identifying features around CpG islands in different cell and tissue types [38]. Features can be divided into different groups. Examples of these feature groups include structural, regulatory (i.e., transcription factor), or those based on the sequence alone (i.e., motifs and "words" in the sequence). The methods and software used in this project for feature generation are outlined below, however any desired software may be added to the job flow.

Thus to summarise this work had the following objective:

To create a scalable, fast, user friendly system for feature matrix generation executed in a timeframe that is realistic by connecting the workflows to a desktop grid.

2 Methods

2.1 Software and Algorithms Used in this Study

2.1.1 BOINC

BOINC is an open-source software that was first developed for the SETI@HOME project and later became available as a general purpose software for projects that need computing power. BOINC can be classified as a client-server system. Many research projects that require intensive computing resources use BOINC as it can be used for volunteer computing and grid computing. It has several subsystems which can be classified into two main groups, client side and server side. BOINC projects are then created to solve specific computational problems. In the past these projects cover many fields of science among them astrology, physics, chemistry, biology, mathematics [39]. Client machines are connected to BOINC using the URL of the project with workunits containing the necessary binaries and input files for achieving a specific task

```
>ExprA_GRCh37_11__70211531_cg25574765_447292_CpG
CACCTTAGACCACAGGAAATGTCTGGTTAACACACGAAGAGATGGAAACGCTCGCAGCCACGCCGCAAACGGTTAGTCACGCC
CCACAGCCTGCACTCCTCCCAGCGCGTTTTCCACTTAAG
```

```
cpg_id banana_bend_structure banana_curve_structure
>ExprA_GRCh37_11__70211531_cg25574765_447292_CpG  16.1475  7.67295
```

Fig. 8 Grid Enable banana input and output files

```
# Output from BTWISTED
# Twisting calculated from 1 to 122 of ExprA_GRCh37_11__70211531_cg25574765_447292_CpG
Total twist (degrees): 4110.2
Total turns : 11.42
Average bases per turn: 10.69
Total stacking energy : -1013.25
Average stacking energy per dinucleotide: -8.37
```

Fig. 9 Btwisted output

and generating output files. Workunit specifications are stored in two xml files which contain a list of input and output files.

The generator program creates workunits, however there are other methods for creating workunits. They can be created by BOINC API, DC – API, Generic master programs or 3G bridge [40]. Further details can be found in the BOINC project website [39]. Running a program under BOINC has some requirements. The binary code of a program should be able to communicate with BOINC, either using API or other techniques which is summarised in Table 1.

Jobs are described in template files which are in XML format and describe input, output and other job parameters. Fig. 1 shows BOINC component relationship both for the client side and the server side. The components in the rectangles are the server side components.

2.1.2 WS-PGRADE/gUSE

gUSE is a distributed computing infrastructure gateway framework which uses the MySQL database management system. It provides the user with the access to the distributed computing infrastructure. WS-PGRADE [41] is a workflow management system implemented as portlets in the Liferay portal and benefits from the features provided by Liferay. From the architectural perspective, these fseatures are web applications deployed in a web container. Liferay itself is an open source portal developed in java which can be deployed in the Tomcat servlet container. Fig. 2 shows the simplified architecture of the system.

2.2 Emboss

EMBOSS is a set of different binaries commonly used to analyze DNA, RNA and protein sequences.

Four applications were used in this case study namely Banana, btwisted, wordcount and Jaspscan. “Banana” can predict bending of a normal DNA double helix and has used by Previti et al. for feature generation. “btwisted” calculates overall twist of the DNA sequence and the stacking energy. Similar to Banana, “btwisted” was used by Previti et al. for structural feature generation [36]. In this study, “Jaspscan” is used to generate a feature matrix based on transcription factor binding sites in the JASPAR database [37 and 42] and “wordcount” is used for generating the number of “words” in the DNA sequence. Each of these programs’ outputs can be modified by an external application to make them usable for classification.

2.3 Hillclimbing

In executing this case study it became apparent that often a further algorithm is needed to search for features in such large data sets. The hill climbing algorithm was therefore employed to the workflow which made use of the BIONC distributed system. It is a method for searching a large search space which cannot be searched exhaustively. It was employed here to identify the best features for classifying a DNA sequence. With this feature subset selection problem, a potential solution is defined by a binary string. The length of the string shows total number of features. Selected features are

cpg_id	btwisted_turn_structure	btwisted_twist_structure	btwisted_stackenergy_structure
>ExprA_GRCh37_11__70211531_cg25574765_447292_CpG	4110.2	11.42	-1013.25

Fig. 10 Btwisted output file

```
CCAC 4
ACGC 3
CACG 3
CAGC 3
CACA 3
CTCC 2
GTTA 2
CGCC 2
GAAA 2
continued.....
```

Fig. 11 wordcount output file.

represented by a one. Non-selected features are represented by a zero. For example, if there is one at a position two it means that feature number two is selected. Since choosing a single starting point can trap the hill climbing algorithm at local optima, the starting point for the algorithm can be any random point in the search space, since different starting points may lead to better solutions. These starting points were run in parallel and the results were compared. Finally the maximum value of the results was selected. Due to the nature of the feature selection problem it was a good candidate for distributing over multiple computing nodes.

2.4 System Design

The BOINC project for this work was created in a Debian Linux 6.0 machine. The applications were ported to BOINC by GenWrapper. The gUSE system was also installed into the same Debian machine and connected to BOINC via the 3 g-bridge. The client machines were seven Microsoft Windows virtual machines on VMware. VMware contained the BOINC client connected to the project. Figure 3 shows client and server side of the test system. Figure 4 shows how the client side subsystems are working with each other. We used a virtual machine for our BOINC clients and this system could be easily extended to use a multicloud system using the methodology described in Previti (2009) [43] if there was a need to scale the local grid system. Similarly as it can be seen in the split application, our data can follow the MapReduce paradigm and the methodology described in Gugnani et al. (2016) [32] which can be used for some part of the workflow we developed. Inside the virtual machine, when a job is available, BOINC calls the application and application calls GitBox to run the shell script provided by the application. Shell script then runs the programs in the virtual machine.

The four EMBOSS applications can be used individually to generate features or merged together if desired. The ultimate goal of the system was to create modular system for feature generation using gUSE.

Each of these applications in the WS-PGRADE portal has three main programs:

Generator: This program generates tasks or workunits which will eventually run on the worker machines. Some examples of these tasks are 1) splitting big files into smaller files 2) Processing large number of files. 3) Generating parameter combinations.

Worker: The workunit generated by generator should be submitted to the worker machines. The worker program processes the workunits and sends the results back. For example finding the maximum of a function given some parameters.

Collector: The collector aggregates results returned from the worker machine. For example, the different parts of a file processed by a worker should merge to give the final result, or in another example finding the

cpg_id	ATCG	TTGC	TTGG	TCTA	TTGA	TCCT	continued
>ExprA_GRCh37_11__70211531_cg25574765_447292_CpG	0	0	0	0	0	1	....

Fig. 12 wordcount output file.

cpg_id	MA0039.2	MA0156.1	MA0173.1	MA0057.1	....
ExprA_GRCh37_11__70211531_cg25574765_447292_CpG	0	1	1	0	...
ExprA_GRCh37_9__140221397_cg13408086_247051_CpG	0	0	0	0	....

Fig. 13 partial view of grid enabled jaspscan output file.

maximum of all returned results from evaluated parameters on the worker machines.

This system provides the facility to add new feature generating elements if desired using the gUSE system. Thus this system can be expanded based on the user demand. This section starts with a description of the simplest form of feature generation with one application and increases by adding all parts to the system. The input to the workflow is a FASTA file containing the DNA sequences with their ID and the output is the feature matrix containing all investigated features. All the DNA sequences were 122 bp in length and flanked the CpG in the DNA sequence See Fig. 5 input file.

2.5 The Grid Enabling of DNA Feature Generation Applications

As mentioned BOINC has been used in many biological projects and is supported by WS-PGRADE/gUSE. It is relatively simple to apply to such projects. The decision to grid enable the work flows was taken in order to increase the computing power available to the user rather than to speed up the execution of the program per se. Many biological and medical data sets are so large that these programs have in the past either been implemented on supercomputers/large servers or implemented in the cloud. The ability to analyse and/or manipulate very large data sets using commonly available non-dedicated desk top computers by non-technical users has been unavailable until now. This is the supplementary issue we are addressing in this work. This study uses the system to generate features from DNA sets for further analysis as a case study of how this strategy can be achieved.

Table 2 This table shows an example of how many features each program generated per file which were then used in the hill climbing search.

Application name	Number of files generated	Number of features
banana	1 per sequence	2
btwisted	1 per sequence	3
wordcount	1 per sequence	256
jaspscan	1 for all sequences	467

The high level algorithm for feature generation is provided in Fig. 6.

There are two FOR loops in the algorithm because running feature generation on each sequence is independent of the other sequences. We can distribute the tasks for these two loops onto separate machines to achieve speedup. The current feature generation programs for the DNA methylation studies use the sequential non-modular method of feature generation. Sometimes this task is done manually in spread sheets and then the feature matrix aggregated as an input to machine learning algorithm. This section provides a modular and distributed example of feature generation. The use of tools like BOINC and gUSE does not influence the design of these kinds of systems, any system with similar properties can be used to achieve these tasks.

The following section provides details of the grid enabling of the four EMBOSS applications in this project. All of these applications had sequences in FASTA format as their input. Outputs are tables with the sequence ids as rows and the columns are the feature names.

2.6 Banana

Banana predicts the bending of a normal DNA double helix [44] and was used for generating two features. These features are in the class “structural features”. Genwrapper [45] was used to port the application into the BOINC desktop grid environment. Banana generates one file for each sequence which has two values (one for bend and one for curve) for each base pair in the sequence (Fig. 7).

The average of these values was used as the feature for the sequence. Gitbox, which is part of GenWrapper supports “awk”, which was used to read the profile files generated by banana to produce data points in table. The two values are extracted from profile files. The program is grid enabled by GenWrapper and BOINC worker machines running this program.

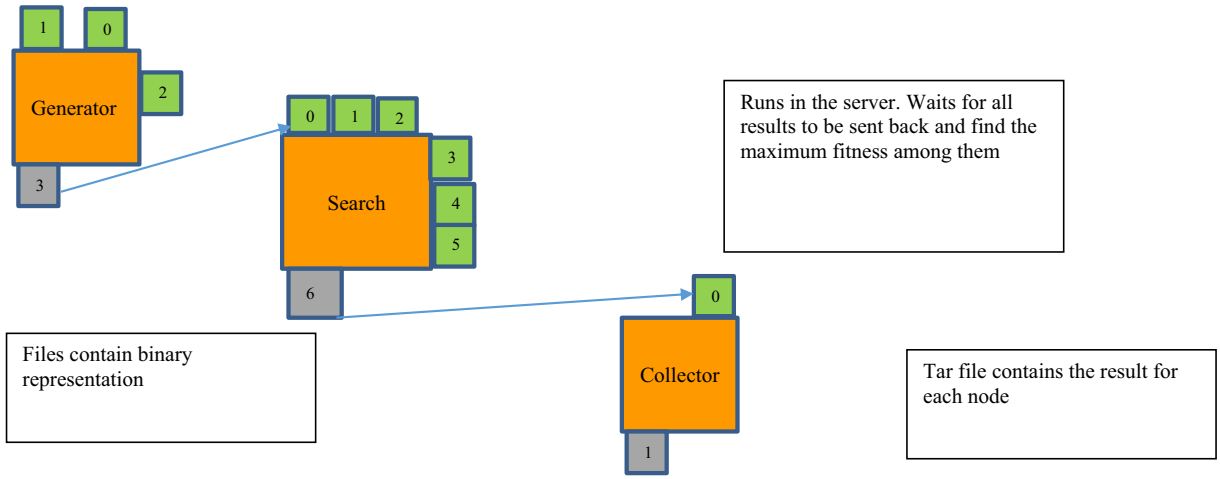


Fig. 14 Hill climbing search workflow graph in the graph editor shows the search node is the worker node, and generator and collector nodes are master nodes

The input for banana program is FASTA file such as that shown in Fig. 8. The output of the banana program is profile file which has value for each base in the sequence. The program extracts these values and makes two features and outputs them in tab delimited file shown in Fig. 8.

2.7 Btwisted

Btwisted is an application which calculates the overall twist of the DNA sequence and the stacking energy [44]. Similar to banana it generates one file per sequence. In each file there are five records: total twist, total turns, average base per turn, total stacking energy and average stacking energy. Figure 9 shows output of standard btwisted program.

Similarly GenWrapper [45] was used to port the application to the BOINC desktop grid computing platform. "awk" was used to read the btwisted file line by line and to generate the three attributes (turn, twist, stackenergy). The grid enabled btwisted output is shown in the Fig. 10.

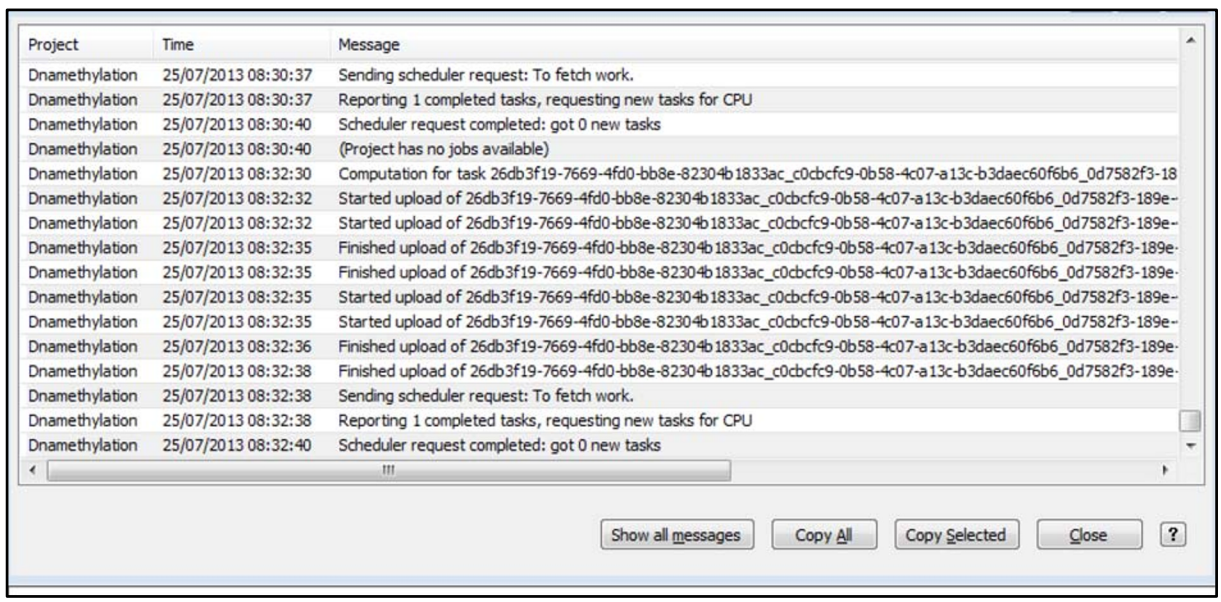


Fig. 15 BOINC client log snapshot shows details of tasks running on client machines

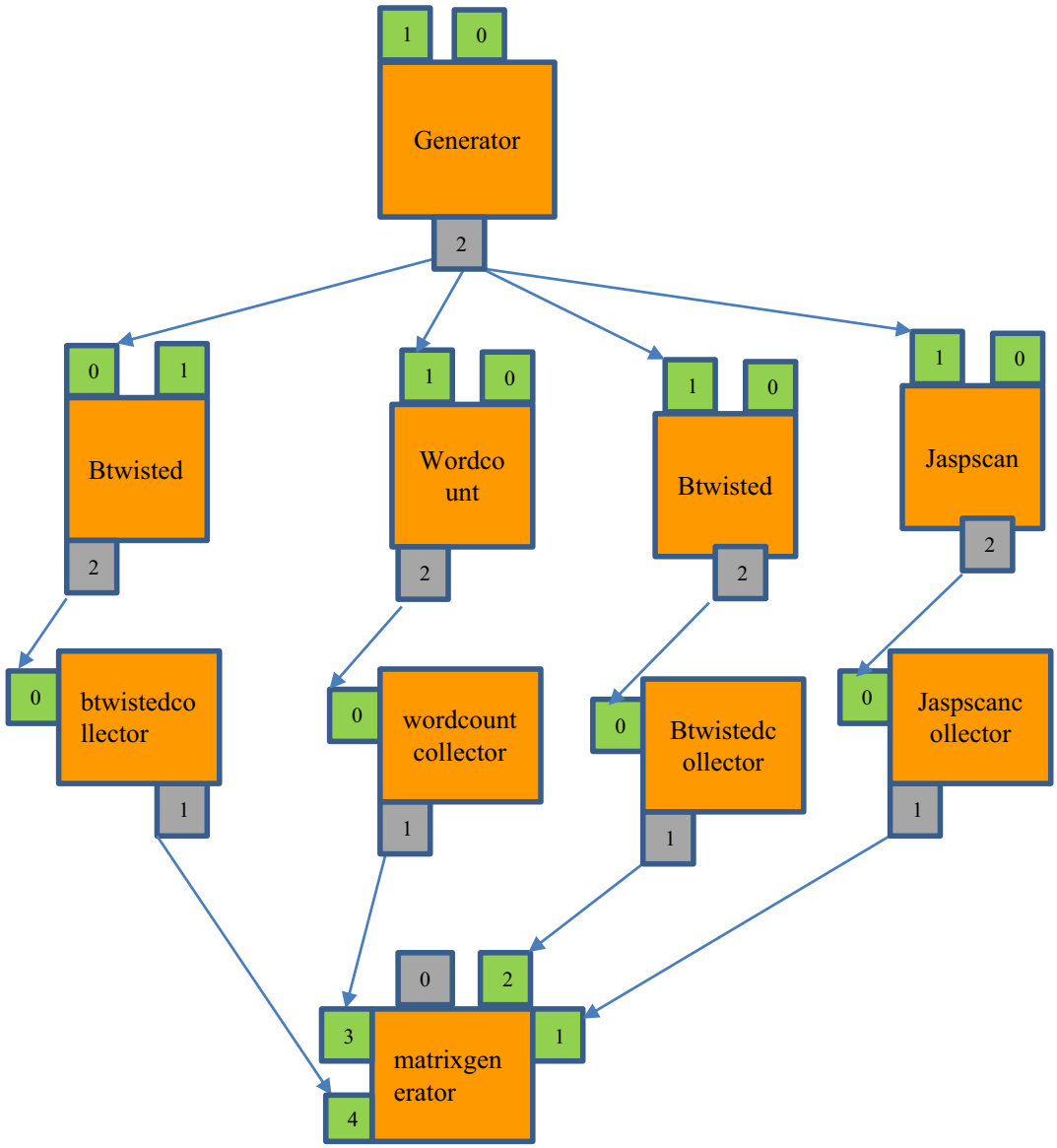


Fig. 16 The completed workflow of the feature matrix generator needs two input files one determining the number of sequences per file and the other is the fasta file of all sequences. Individual application results could be downloaded as in the workflow they were conFig.d as permanent. The final results were generated in the matrix generator node

2.8 Wordcount

Wordcount counts the number of times a string pattern or word occurs in each sequence. Words are made of base pairs of sequence of specific size. It moves along the sequence and counts the words. It generates one file per sequence similar to banana and btwisted. In each file there are two columns one is the word name and the other is the number of occurrences of that word. CCAC occurred 4 times in our input sequence. An example of a file is shown in Fig. 11.

First the shell script program generates all possible patterns of the string of 4 characters A, C, G and T. It then used these files to generate a feature matrix file containing the frequency of each word in the sequence. GenWrapper was used here to port the application to the BOINC. The output of the grid enabled wordcount is shown in the Fig. 12.

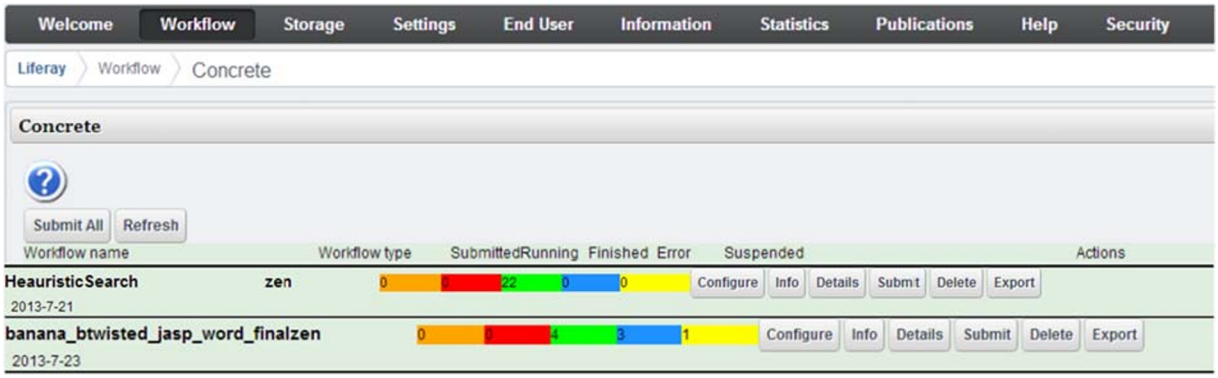


Fig. 17 an example of a job Submission page, the "conFig." button should be used to add input and output files and to define binaries for each workflow.

2.9 JaspScan

Jaspscan scans the sequence for the motifs listed in the JASPAR transcription factor motif database [46]. This database contains the collection of known transcription factor consensus binding sites and is collated from published papers and has an open data access policy. It generates one file for all sequences in the input FASTA file.

Jaspscan generates one file for all sequences, unlike other ported applications which generates one file per sequence. Here we only need to find the section that has the information for each sequence, to generate the matrix file. An example output is shown in the Fig. 13. Here each section is marked with "#Sequence". The string "#===" is used to distinguish between different sequences. A full list of transcription factor binding motifs was provided to the program. In the example in Fig. 13 the value of the feature MA0156.1 is 1 because there is one entry in the matrix.

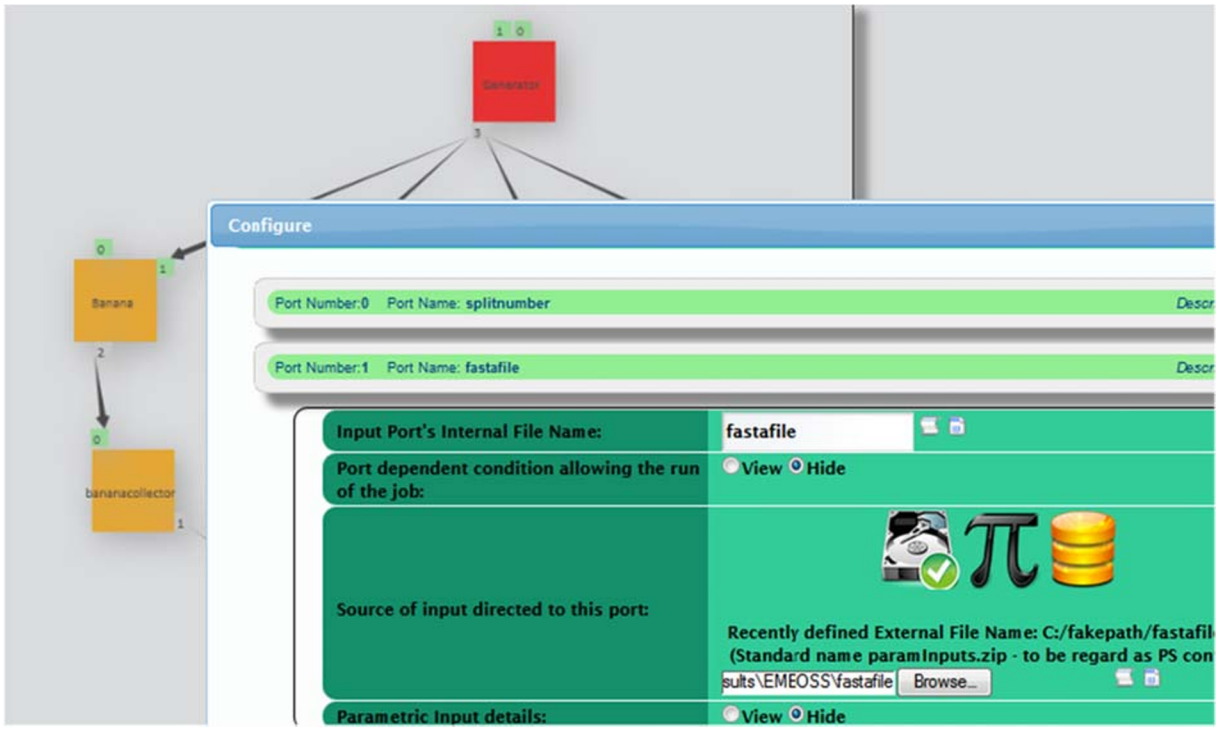
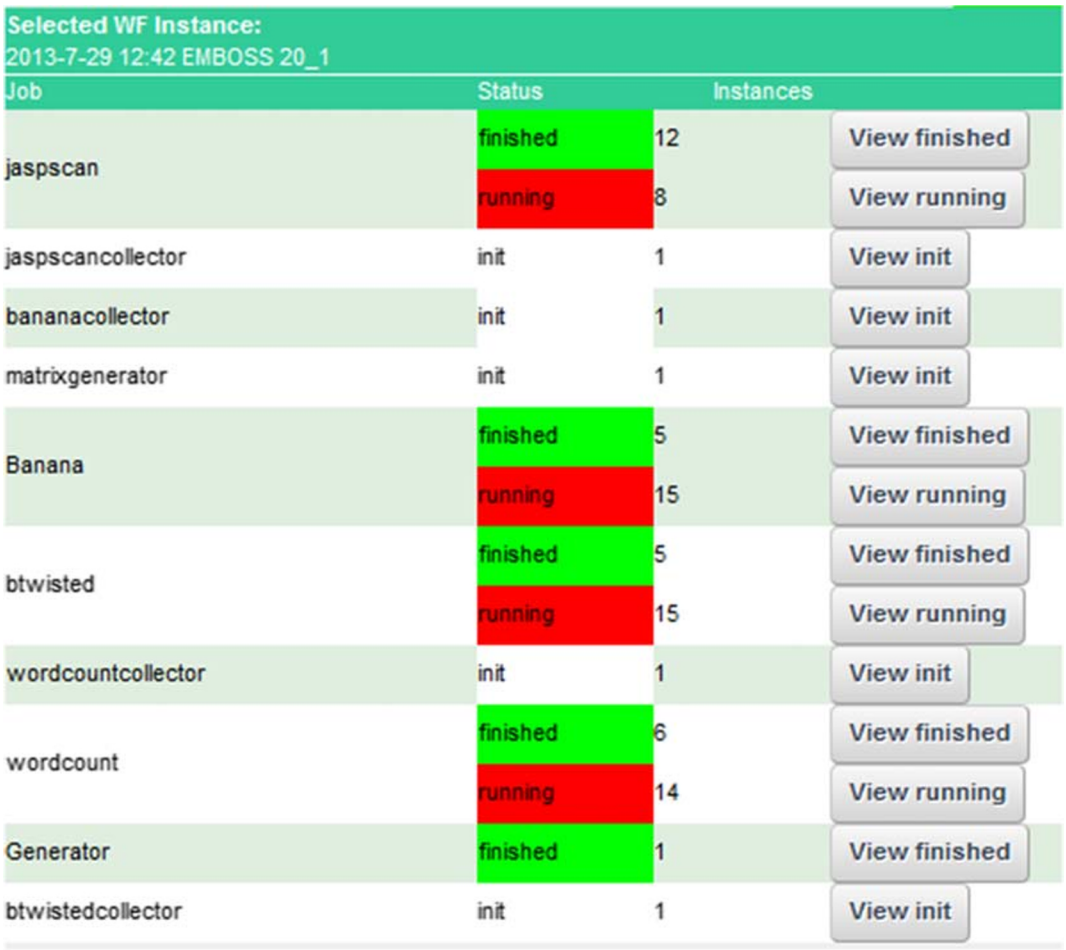


Fig. 18. an example of Workflow configuration page.



Selected WF Instance:
2013-7-29 12:42 EMBOSS 20_1

Job	Status	Instances	
jaspscan	finished	12	View finished
	running	8	View running
jaspscancollector	init	1	View init
bananacollector	init	1	View init
matrixgenerator	init	1	View init
Banana	finished	5	View finished
	running	15	View running
btwisted	finished	5	View finished
	running	15	View running
wordcountcollector	init	1	View init
wordcount	finished	6	View finished
	running	14	View running
Generator	finished	1	View finished
btwistedcollector	init	1	View init

Fig. 19 Details of the status of each job after workflow submission could be monitored in ws-pgrade portal. This Fig. shows the workflow submission for an input file which contained 20 sequences. It then indicated that each sequence should be submitted individually so eventually 20 jobs were submitted

The program extracts this information for each sequence from Jaspscan output file.

Each application in the work flow generates a file with numerous features in each file as exemplified in Table 2.

2.10 Grid Enabled Hill Climbing Search

The Hill Climbing program was developed as a Java program, and similar to the previous section, GenWrapper was used to call the Java program on the worker machines and also to zip the results in the case of batching more than one binary in a job. The inputs consisted of two files; one file contained the string representations of the starting point solutions, whilst the second file contained the parameters given to the program. These parameters defined the number of iterations, classification method, measurement index and the name of the file that contained starting point's information.

The collector application was a shell script that runs on the server, which extracted the files received; each received file then contained two files. The first is the "<FILENAME> Max", each line of the file contained information about when the maximum value is updated in the hill climbing search. Each line in the file <FILENAME> contained the calculated measurement and the binary representation of the solution in each iteration. Then the script searched all the "<FILENAME> Max" files to find the maximum in those files. It is then copied this file into a folder that contained all files. It then tar zipped all the files together

1577	1152	Unsent [2]	Init [0]	New [0]	Initial [0]	Initial	0	----	0
1576	1151	In Progress [4]	Init [0]	New [0]	Initial [0]	Initial	0	8 (mmgh)	0
1575	1150	Over [5]	Success [1]	Uploaded [5]	Valid [1]	Deleted	0	5 (mmgh)	30
1574	1149	In Progress [4]	Init [0]	New [0]	Initial [0]	Initial	0	3 (mmgh)	0
1573	1148	Over [5]	Success [1]	Uploaded [5]	Valid [1]	Deleted	0	3 (mmgh)	30
1572	1147	In Progress [4]	Init [0]	New [0]	Initial [0]	Initial	0	7 (mmgh)	0
1571	1146	Over [5]	Success [1]	Uploaded [5]	Valid [1]	Initial	0	3 (mmgh)	30

Fig. 20 BOINC result page for job progress

as the final result. It used the awk program to search through each line and the maximum field of the file.

In this project the gUSE portal workflow system was also applied to submit hill climbing jobs to BOINC. First the workflow topology was defined in the graph editor of the workflow. Each port in our example is the representation of the file and each node are the applications running in the system. The arrows between the

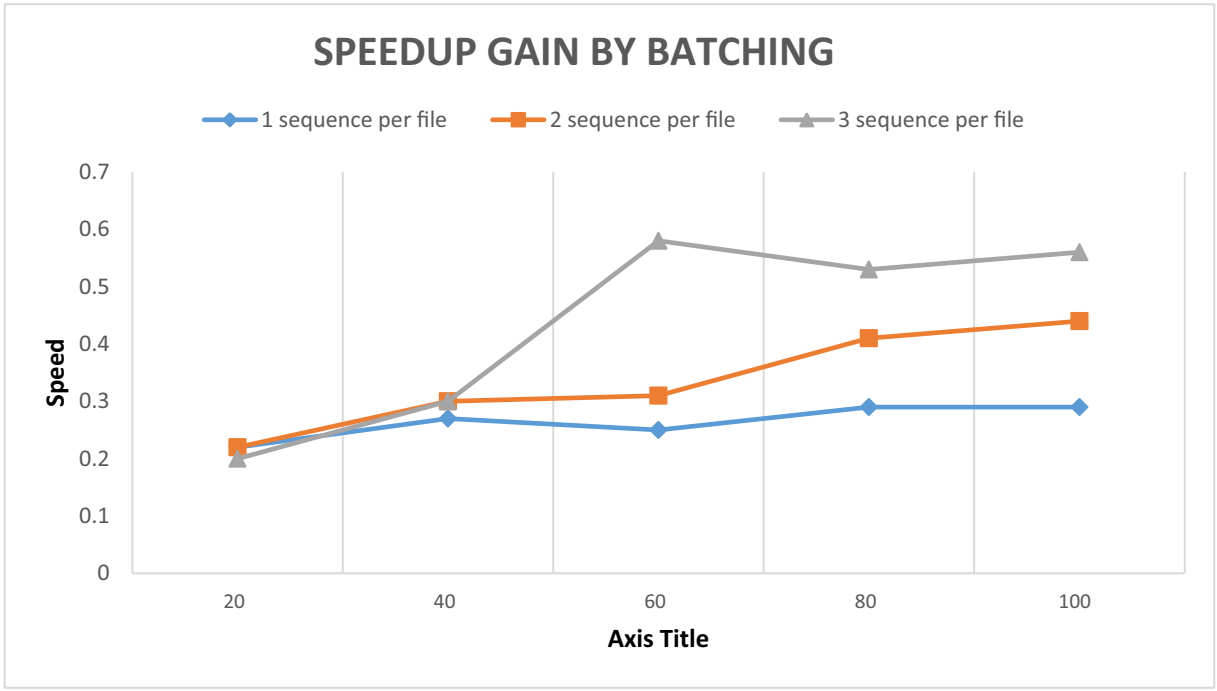


Fig. 21 speedup improvement by batching of more than 1 sequence per file.

number of sequences	1 sequence per file	2 sequences per file	3 sequences per file
20	0.22	0.22	0.20
40	0.27	0.30	0.30
60	0.25	0.31	0.58
80	0.29	0.41	0.53
100	0.29	0.44	0.56

Fig. 22 Performance results for different number of sequences analysed and different number sequence per files

ports represent file transitions. Green ports are input ports and grey ports represent output ports. Fig. 14 shows a completed workflow.

Generator and Collector applications were running in the server as mentioned in the previous section. In another setting it is possible to use another machine other than server to run these programs as long as they are supported by the gUSE resources for the nodes.

The workflow configuration for the hill climbing search was similar to the EMBOSS application i.e. for the search node with the parameter file and for the Generator node with the "rankedlisttodistribute" file. After submitting jobs clicking the "details" button should indicate that it is running.

The log file of BOINC manager in the worker machine can indicate the progress of the works in the client side. The following snapshot shows the connection of the client to the BOINC project and one completed task. (Fig. 15).

3 Results

3.1 Single Application Feature Generation

The banana application was the first program to be used for feature generation in the workflow. A workflow graph was created in the workflow editor. The generator split the sequence file and then it generated a defined number of sequences per file. The number of sequences per file was determined in "splitnumberfile". Banana was executed for each file in worker nodes and generated "tabdelimited" files similar to the matrix mentioned in the previous section. The collector pasted each new file, one after another, except the header of the file, and creates new file. Because the program created the same headers for each file in the same order, this does not create any inconsistency between the values.

3.2 Multiple Application Feature Generation Workflow

Adding btwisted to this workflow needed the additional step of combining the results of the banana collector with the results of the btwisted collector. The feature matrix headers were in order, so each application had its own collector. The source code for the collectors is similar, but they were used in separate nodes. This made it possible to download the results for each individual EMBOSS application. Because final results should be in order, they should be sorted by name in the collector node.

At this stage any other application could be added to the workflow using the same guidelines, without making changes to other parts of the workflow. Similarly, parts that are not required can be removed from workflow. The generator node and matrixgenerator node will be the same for all nodes. The final workflow designed in this work is shown in the Fig. 16.

Fig. 23 speedup results for different number of iterations of hill climbing for 20 to 100 iterations.

number of jobs	number of classifier calls	speedup
20	400	1.781782531
40	800	2.33002331
60	1200	2.672673797
80	1600	2.817702607
100	2000	2.442766373

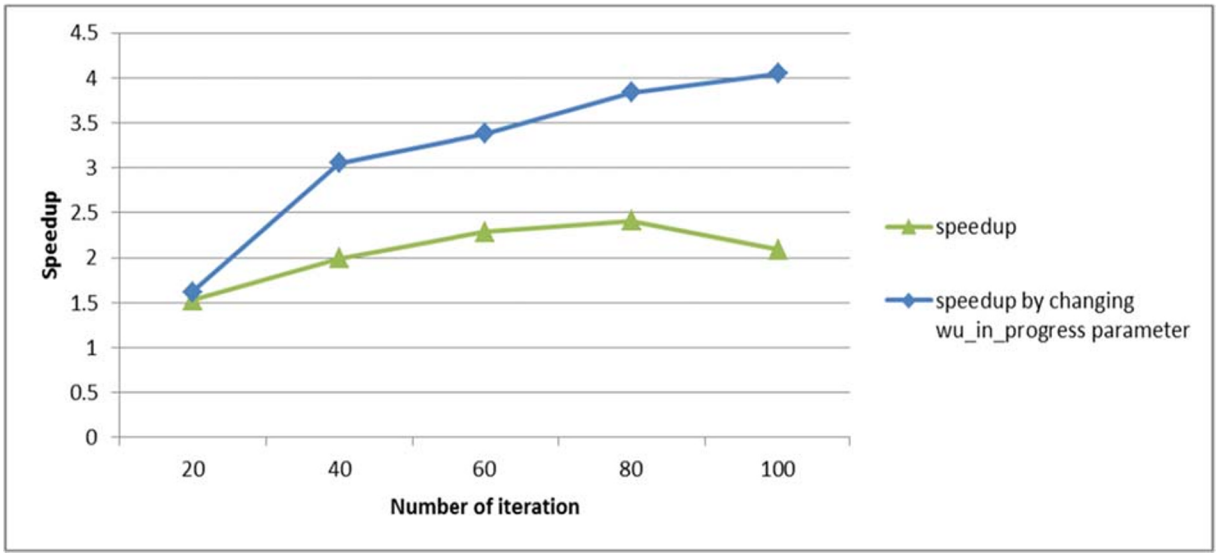


Fig. 24 The graph shows speedup results for two different kinds of configuration. Improvement could be seen as the number of "wu_in_progreess" was restricted

MatrixGenerator simply pasted all the files together. It used the dot product capability of the gUSE system to collect all the files.

paste allresult_* > matrixfeaturefile.

Workflows were submitted via the portal liferay/ workflow/concrete tab Fig. 17. By choosing the concrete tab a list of generated workflows could be seen. In this page users can overview the overall status of the workflows and take appropriate actions. For each new instance of the workflow new files could be uploaded to the system. This task can be done by choosing the conFig. button in the actions section. The two input files (number of sequence per file and FASTA file) could be uploaded in this page Fig. 18.

The details of each workflow node progress could be examined in a page similar to that shown in Fig. 19. Whenever the workflow nodes finished their job, the results could be downloaded by clicking view all the "content" buttons.

Another way to check the status of jobs was by querying the BOINC database. The details of the jobs running in the BOINC workers and whether or not they were sent to workers can be checked by querying the BOINC database. This can be done by examining the admin webpage of the BOINC project. (Fig. 20) Whenever the valid results were sent back to the BOINC server, 3Gbridge daemon uploaded them to the upload folder and cancel the workunits.

3.3 Performance Testing

Whenever there are a large number of sequences, in the case of whole genome for example, the running time of

Fig. 25 Shows the comparison of speed-up for different numbers of jobs and different configurations

number of jobs	speedup	speedup by changing wu_in_progress parameter
20	1.524378	1.619652152
40	1.993418	3.048756992
60	2.286568	3.380143622
80	2.410645	3.839175472
100	2.089874	4.04913038

Fig. 26 details of speedup as programs progress.

percent of work completed	J48	SVM	naïve bayes
10	0.973529759	0.985894581	0.9747651
20	1.843049929	1.937272064	1.93653333
30	2.741146292	2.801687764	2.82935065
40	3.586475537	3.719887955	3.67696203
50	4.400751871	4.620737648	4.5672956
60	5.144417694	5.301397206	5.37925926
70	5.931072266	5.781094527	5.37925926
80	4.073423219	3.900146843	3.8990604
90	4.138020416	4.378021978	4.37177258
100	4.441541538	4.754744003	4.84133333

feature generation will increase dramatically. In order to assess the running time of the Humethylation450k Illumina platform data used in this case study, which has data for nearly half a million DNA sites, we can estimate the running time: Each job, when all feature generation programs run takes nearly 68 s (wordcount+ banana+ btwisted+ jaspscan) on Intel Core(TM) 2 Duo CPU E7400 2.8 GHz. For half million sequences and 2 million work units (without batching), it would take nearly one year (393 days) to run on one such machine. Depending on the number of available similar machines we can get the result much faster. Thus any decrease in running time would be of considerable interest to users. We achieved an overall running time for the best experiment for 653 sequences of 148 min. This is how we did this using a very modest number of computers:

We had seven such machines at our disposal (CPU spec Intel Core(TM) 2 Duo CPU E7400 2.8 GHz) running Microsoft Windows 7 operating system in a LAN network. Analysis of the initial tests of the grid enabled workflow showed that because of the short running time of each sequence there might be a communication overhead. For example, 100 sequences and one sequence per file will cause 400 work units submissions for all the algorithms. Therefore increasing the number of sequences per file may improve the performance. To see if this improvement can be realized using this strategy we tested the workflow performance for the feature

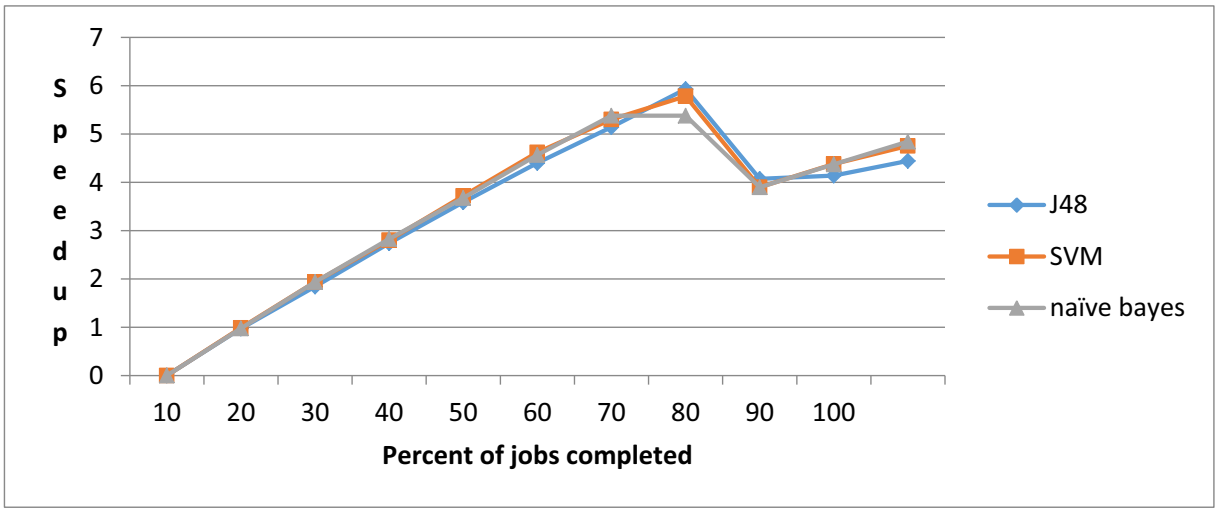


Fig. 27 shows speedup results for different classification methods in feature subset selection.

searches whilst increasing the number of the sequences in each FASTA file. Fig. 21 and Fig. 22 shows the speed improvement by the batching of more than one sequence per file. The graph illustrates that by placing only 2 or 3 sequences per file instead of having only one we could achieve better performance. Thus the batching was proven to reduce the large number of jobs submitted reducing the communication overhead. Therefore by putting the sending and receiving jobs together in this way we can reduce the slowdown.

To observe the effect of running the hill climbing part of the workflow on a volunteer grid initial tests on the hill climbing search were done with gradually increasing iterations for the same number of jobs, measuring the performance of the system. In order to compare the run time with the sequential time, average computation time of one iteration of a model for 100 iterations was calculated. The initial performance test was done using the naïve Bayes classifier. Fig. 23 shows the performance results for different number of iterations of hill climbing for 20 to 100 iterations. This data shows that the speedup did improve a little as jobs became more computationally intensive. The initial performance test gave nothing better than 2.8 times faster than sequential times, in the best case, with further investigation of the submitted jobs, it was realised that all jobs were assigned to a few available hosts, and some hosts didn't get any jobs. In order to improve this situation, the number of jobs per host was limited. This can be done in BOINC by setting the parameter "wu_in_progress" in the BOINC configuration file. Applying this restriction to one job in progress for each host further improved performance. The results can be seen in the Figs. 24 and 25.

After initial testing, the system was used for the hill climbing search on each classification method. The results showed that this system achieved a maximum of 5.37 speedup out of a possible 7, since we had 7 machines in our test environment. The difference between the speedup results should be seen as communication overhead and run times of collector and generator nodes. Results are shown in Fig. 26 and Fig. 27.

4 Conclusion

This report provides details of the preparation and testing of a system for the tasks of feature generation and feature subset searching on a volunteer desktop grid in order to generate feature matrices. This work had two objectives:

a) The first objective was to create a scalable, fast, user friendly system for feature matrix generation.
b) The second objective was to accelerate feature generation and feature subset selection to a degree that made analysis possible in a timely manner.

The first objective has been completed by creating a user friendly workflow using gUSE. The workflow provided modular feature generation, allowing other software to be added to the work flow with ease. Furthermore in this system the addition of new feature generating nodes did not change the whole system.

With regard to the second objective, connecting the small number of the jobs in the feature generating workflow to BOINC did not in itself provide any improvement in performance; reduction in slowdown seen when a single desktop computer was achieved when we used BIONC coupled to the workflow to analyse an increased number of sequences to the quantity more commonly analyzed in genomic big data projects. To further achieve this improvement and reduce communication time, the number of sequences per work unit was increased. The grid enabled hill climbing search shows better performance by connecting to the BOINC platform in the test environment. Thus we show that the BOINC system is a good tool to accelerate the execution time of feature generation and feature subset selection in large data sets.

Acknowledgements We would like to thank the technical and administration staff of the Department of Computer Science, Brunel University for their support and Brunel University for support in kind.

Author Responsibilities MG designed and implemented the system, AP provided and designed the biological problem and advised on the needs of biologists, ST advised on the distributed system and workflow design, SS advised on the hill climbing technique.

Compliance with Ethical Standards

Competing Interests The authors declare that they have no competing interests.

References

1. Jordan, M.I., Mitchell, T.M.: Machine learning: trends, perspectives, and prospects. Science. **349**(6245), 255–260 (2015)
2. Q Zou, L Chen, T Huang, Z Zhang and Y Xu Machine Learning and Graph Analytics in Computational Biomedicine. *Artificial Intelligence in Medicine* **83**, November, Page 1 and papers therein; (2017)
3. I.H. Witten, E. Frank, M.A. Hall and C.J. Pal, Data Mining: Practical machine learning tools and techniques. (Morgan Kaufmann 2016)
4. W. Cheng, G. Kasneci, T. Graepel, D. Stern and R. Herbrich Automated feature generation from structured knowledge. In Proceedings of the 20th ACM international conference on Information and knowledge management (pp. 1395–1404). ACM. (2011)
5. H. Paulheim and J. Fümkranz June. Unsupervised generation of data mining features from linked open data. In Proceedings of the 2nd international conference on web intelligence, mining and semantics (p. 31). ACM. (2012)
6. L. Friedman and S. Markovitch Recursive Feature Generation for Knowledge-based Learning. arXiv preprint arXiv:1802.00050. (2018)
7. Menezes, J.A., Cabral, G., Gomes, B.T.: Genetic algorithms for feature generation in the context of audio classification. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering. **10**(2), 427–430 (2017)
8. Afgan, E.; Baker, D.; van den Beek, M.; Blankenberg, D.; Bouvier, D.; Čech, M.; Chilton, J.; Clements, D.; Coraor, N.; Eberhard, C.; Grüning, B.; Guerler, A.; Hillman-Jackson, J.; Von Kuster, G.; Rasche, E.; Soranzo, N.; Turaga, N.; Taylor, J.; Nekrutenko, A.; Goecks, J. (8 July 2016). "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic Acids Res. 44 (W1): W3–W10
9. Johannes Köster and Sven Rahmann. "Snakemake - A scalable bioinformatics workflow engine". Bioinformatics 2012
10. J Gray. Jim Gray on eScience: A transformed scientific method. In The Fourth Paradigm: Data-Intensive Scientific Discovery, Tony Hey, Stewart Tansley, and Kristin Tolle (Eds.). (Microsoft, xix–xxxiii. 2009)
11. Hey, T., Tansley, S., Tolle, K. (eds.): The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research (2009)
12. Kell D B and Oliver S G. Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era. *BioEssays* **26**, 1, DOI:https://doi.org/10.1002/bies.10385 (Jan. 2004)
13. Gorton, I., Greenfield, P., Szalay, A., Williams, R.: Data-intensive computing in the 21st century. Computer. **41**(4), 30–32 (2008)
14. Deelman E, Vahi K, Rynge M, Juve G, Mayani R, and Ferreira da Silva R. Pegasus in the cloud: science automation through workflow technologies. *IEEE Internet Comput.* **20**, 1, 70–76. DOI:https://doi.org/10.1109/MIC.2016.15 (Jan. 2016)
15. Kacsuk, P., Kecskemeti, G., Kertesz, A., et al.: Infrastructure Aware Scientific Workflows and Infrastructure Aware Workflow Managers in Science Gateways J Grid Computing. **14**, 641 (2016) https://doi.org/10.1007/s10723-016-9380
16. Wassenaar, T.A., van Dijk, M., Loureiro-Ferreira, N., et al.: WeNMR: Structural Biology on the Grid J Grid Computing. **10**, 743 (2012) https://doi.org/10.1007/s10723-012-9246-z
17. M. McLennan, R. Kennell, "HUBzero: a platform for dissemination and collaboration in computational science and engineering," *Computing in Science and Engineering* 12(2), pp. 48–52, March/April, 2010
18. Kacsuk, P., Farkas, Z., Kozlovszky, M., et al.: WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities J Grid Computing. **10**, 601 (2012) https://doi.org/10.1007/s10723-012-9240-5
19. Deelman, E.: Grids and clouds: making workflow applications work in heterogeneous distributed environments. International Journal of High Performance Computing Applications. **24**(3), 284–298 (Aug. 2010) https://doi.org/10.1177/1094342009356432 2010
20. Kacsuk P (Ed.). Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities. DOI: https://doi.org/10.1007/978-3-319-11268-8 (2014)
21. Liew C S, Atkinson M P., Galea M, Ang T F, Martin P, and Van Hemert J I. Scientific workflows: moving across paradigms. *ACM Comput. Surv.*. **49**, 4, Article 66 DOI: https://doi.org/10.1145/3012429 (December 2016)
22. Kacsuk, P.: P-GRADE portal family for grid infrastructures. *Concurrency and Computation: Practice and Experience* Special Issue: IWPLS 2009. **23**(3), 235–245 (2011)
23. Balasko, A .: Workflow Concept of WS-PGRADE/gUSE. Science Gateways for Distributed Computing Infrastructures:Development Framework and Exploitation by Scientific User Communities, pp. 33–50 doi:https://doi.org/10.1007/978-3-319-11268-83 (2014)
24. S.C. Shah Recent Advances in Mobile Grid and Cloud Computing. *Intelligent Automation & Soft Computing*, pp.1–13. (2017)
25. Ellert, M., et al.: Advanced resource connector middleware for lightweight computational grids. Futur. Gener. Comput. Syst. **23**, 219–240 (2007)
26. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the condor experience. Concurrency and computation: practice and experience. **17**(2–4), 323–356 (2005)
27. Foster, I.: Globus toolkit version 4: software for service-oriented systems. IFIP international conference on network

and parallel computing, Springer-Verlag LNCS. **3779**, 2–13 (2005)
28. David, P.: Anderson: Public Computing: Reconnecting People to Science. Conference on Shared Knowledge and the Web, Residencia de Estudiantes, Madrid, Spain (2003)
29. , et al.: The DECIDE science gateway. J Grid Comput. **10**, 689–707 (2012). https://doi.org/10.1007/s10723-012-9242-3Ardizzone, V., Barbera, R., Calanducci, A. et al.: The DECIDE science gateway. J Grid Comput **10**, 689 doi:https://doi.org/10.1007/s10723-012-9242-3 (2012), 707
30. Costa, A., Massimino, P., Bandieramonte, M., et al.: An innovative science gateway for the Cherenkov telescope array. *J Grid Comput*. **13**, 547 (2015). https://doi.org/10.1007/s10723-015-9330-2
31. R. Grunzke, J. Krüger, R Jäkel., et al.: Metadata Management in the moSGrid Science Gateway – Evaluation and the Expansion of Quantum Chemistry Support. *J Grid Computing*. doi:https://doi.org/10.1007/s10723-016-9362-2 (2016)
32. Gugnani, S., Blanco, C., Kiss, T., Terstyanszky, G.: Extending science gateway frameworks to support big data applications in the cloud. Extending science gateway frameworks to support big data applications in the cloud J Grid Computing. **14**, 589–601 (2016). https://doi.org/10.1007/s10723-016-9369-8
33. Farkas, Z., Kacsuk, P., Hajnal, Á.: Enabling workflow-oriented science gateways to access multi-cloud systems. Journal of Grid Computing. **14**(4), 619–640 (2016)
34. C.M. Taylor BOINC user stats https://boincstats.com/en/stats/-1/user/detail/3531367/overview accessed 9/9/2016
35. Bazinet, A.L., Cummings, M.P.: Subdividing long-running, variable-length analyses into short. Fixed-Length BOINC Workunits J Grid Computing. **14**, 429. https://doi.org/10.1007/s10723-015-9348-5–441 (2016)
36. F. Gutierrez, D. Azevedo, M. Barreto and R. Zucoloto Support for bioinformatics applications through volunteer and scalable computing frameworks. In *Cluster Computing (CLUSTER),* 2014 *IEEE International Conference* (pp. 364–370). IEEE. (2014)
37. Cook, C.E., Bergman, M.T., Finn, R.D., Cochrane, G., Birney, E., Apweiler, R.: The European bioinformatics institute in 2016: data growth and integration. Nucleic Acids Res. **44**(D1), D20–D26 (2015)
38. M. Ghorbani, M. Themis, A. Payne Genome wide classification and characterisation of CpG sites in cancer and normal cells. *Comput Biol Med.* **1**;68:57–66. doi: 10.1016/j.compbiomed.2015.09.023. Epub 2015 Oct 23. (2015)
39. BOINC 2017 https://boinc.berkeley.edu/ accessed 12/09/2017
40. Marosi, A., Kovács, J., Kacsuk, P.: Towards a volunteer cloud system. *Futur. Gener. Comput. Syst.* **29**(6), 1442–1451 (2013)
41. Kacsuk, P., Farkas, Z., Kozlovszky, M., Hermann, G., Balasko, A., Karoczkai, K., Marton, I.: WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities. *Journal of Grid Computing*. **10**(4), 601–630 (2012)
42. C.B. Ries, C. Schroder and V. Grout Approach of a UML profile for Berkeley Open Infrastructure for network computing (BOINC), *Computer Applications and Industrial Electronics (ICCAIE),* 2011 *IEEE International Conference*, pp. 483. (2011)
43. Previti, C., Harari, O., Zwir, I., del Val, C.: Profile analysis and prediction of tissue-specific CpG island methylation classes. *BMC Bioinformatics*. **10**(1), 116 (2009)
44. Rice, P., Longden, I., Bleasby, A.: EMBOSS: the European molecular biology open software suite. Trends Genet. **16**, 276–277 (2000)
45. A.C. Marosi, Z. Balaton and P. Kacsuk GenWrapper: a generic wrapper for running legacy applications on desktop grids, *Parallel & Distributed Processing,* 2009. *IPDPS 2009. IEEE International Symposium on* IEEE, pp. 1. (2009)
46. Jaspar 2017, http://jaspar.genereg.net/ accessed 12/09/2017