A Grid-Enabled Problem Solving Environment for QTL Analysis in R

Mahen Jayawardena[1], Carl Nettelblad[1], Salman Zubair Toor[1], Per-Olov Östberg[2]
Erik Elmroth[2], Sverker Holmgren[1]
[1] Department of Information Technology, Uppsala University, Box 337, SE-751 05, Uppsala, Sweden
E-mail: (mahen, salman, carl.nettelblad, sverker)@it.uu.se
[2] Department of Computing Science and HPC2N, Umeå University, SE-901 87, Umeå, Sweden
E-mail: (p-o,elmroth)@cs.umu.se

Abstract

We present a grid-enabled problem solving environment (*PSE*), for multidimensional QTL analysis. The concept of a computational grid has not been fully realized within this and other application fields, mainly due to the need for significant rearchitecting of problem-specific code bases. Our environment allows transparent and efficient utilization of available heterogeneous computational resources. The PSE is built around the open source statistical software system R.

The PSE can be used as a grid-enabled workflow system by the end-users, which is intended to be biologists with a pragmatic approach to high performance computing. Use of the *Grid Job Management Framework* (GJMF) offers the PSE an abstractive and reliable job management interface that supports multiple grid middleware implementations. The use of R enables a tight workflow between computationally demanding analysis and further visualization or processing steps taking place locally, or even interactively, on the user's desktop. We have developed a prototype of such a system using existing components. The architecture has been chosen to ensure transparent use of different computational resources, even from an application developer perspective, by serializing the full state for a request and sending it to a different computational node for deserialization and execution. Based on this prototype, we illustrate some example workflows and timing results from actual multidimensional searches done on experimental data, which are now possible to complete within an hour rather than over the course of several days.

1 Introduction

A computational grid is a combination of computer and data storage infrastructure that, utilizing data communication networks and software frameworks (grid middlewares), delivers computing and data storage services based on distributed hardware and software resources [20]. Grids enable communities to share geographically distributed and technically diverse resources for solving problems that can pose high demands on computational throughput and data storage capacity.

Bioinformatics has been an early adopter of computational grids, e.g. within the EMBRACE project [2] and several TeraGrid Science Gateways [7]. However, the use of distributed computing and storage resources by the bioinformatics community could be further increased if a wider set of tools more resembling the currently used computing environments were available to support their research. The number of problems within the field of bioinformatics that can benefit from using distributed resources is diverse; some are data-dominated, some are dominated by massive-scale computing for analysis, and some are a combination of the two. One area with emerging large-scale computing needs is analysis of quantitative traits, *QTL analysis*. The goal of such analysis is to determine a set of loci in the genome that are responsible for the genetic component of a quantitative trait, using phenotypic and genotypic data gathered from an experimental population. The analysis uses a powerful statistical model, and the computational requirements rapidly increase when several loci are searched for simultaneously (to find interacting loci and/or to better assess the individual effects). In QTL analysis, randomization tests are also performed to determine empirical significance thresholds, resulting in a large number of independent tasks that can be naturally distributed in a computational grid infrastructure.

Current grid middlewares expose complex inner architectures of the distributed computing environment to the application developers, something which can make design and implementation of parallel applications for efficient use of grid resources a time-consuming and challenging task. The need for grid users to adopt new usage models, e.g., job description languages and parallel programming models, further complicates this process. To remedy this situation, much research has gone into making the resources available in a more accessible form. Grid portals have been one such solution [4, 7, 37], and grid-enabled problem solving environments (*PSE*) present another route forward [9, 40]. A grid-enabled PSE makes the bookkeeping and management tasks related to utilizing the grid invisible to the user and enables the computationally critical tasks to be deployed on the grid with an efficient and automatic distribution of data and computational tasks as well as aggregation of final results. Such an environment can also provide the user with a rich array of specifically designed tools, e.g. to aid in pre- and postprocessing and visualization.

In this paper, we present an integrated, grid-enabled PSE for performing multidimensional QTL analysis, realized by using the open source statistical software system R [5] as a "workflow engine". This approach is similar to the one taken in the GridSolve project [40], where e.g. Matlab is used for describing the workflows and a grid-based numerical linear algebra solver can be transparently utilized. We have chosen the R platform as it is widely used by biologists active in quantitative genetics, noting that several R packages aimed at this field have already been developed, see e.g. [12]. However, these packages perform all computations on the local system, resulting in limitations on the complexity of the problems that can be studied. We have developed a prototype system where multidimensional QTL analysis tasks can be performed on external resources, e.g. a computational grid, from within the R environment. Pre- and post-processing can then be performed within R where also the calls to the QTL search modules are included. Furthermore, using standard R programming and possibly earlier developed R routines and tools, complete workflows for advanced QTL analysis settings can be easily implemented. Our PSE is not meant to be a complete generic workflow system, but rather present a tailor-made tool for the specific needs connected to multidimensional QTL analysis. The PSE supports the execution of demanding QTL searches across distributed computing resources in the grid. The PSE also has the capability to perform randomization testing where independent searches within each randomization test are deployed on different grid nodes.

A main feature of the approach taken is that it enables the user to transparently employ different types of computational resources for different aspects of the total QTL analysis workflow. Using identical calls to a computational routine, the computations can e.g. be performed on the users local system, on a cluster at the local computing center, or on a distributed grid system, depending on actual needs and resource availability at the time of execution.

The research presented in this paper builds on previous work [25–27], where we studied how QTL analysis computations can be performed on the grid and other parallel computing resources such as clusters and shared memory servers. In these earlier papers, QTL analysis was performed as a stand-alone process where the raw data and the results could not be analyzed without using external software, and several steps had to be performed "manually". For example, genetic marker data first had to be preprocessed to result in probabilities for line origin in a mesh of closely spaced genome positions, which were then probed in the QTL search.

The rest of this paper is organized as follows: We start by describing the genetics application in Section 2. The architecture of our PSE is discussed in Section 3 and the proof-of-concept implementation is discussed in Section 4. Some conclusions and future work are discussed in Section 5.

Table 1: *Runtime (s).*

Jobs (p)	d=3	d=4	d=5
1	32	331	3749
2	24	221	2245
4	13	222	1628
8	10	142	1104
16	9	78	838
32	-	79	849
64	-	-	602

2 Background

The QTL mapping code used in this paper employs a model where d QTL affect the trait under study. A set of d locations in the genome can be identified by the vector $\mathbf{x} = [x_1 \quad x_2 \ldots x_d]$, where $x_i \in [0, G]$, G being the size of the genome (normally measured in centi-Morgans [cM]). Let n be the number of individuals in the population, $\mathbf{y}$ the vector of n phenotype observations, $k \geq d + 1$ the total number of parameters in the model, and $\mathbf{b}$ the vector of k effects. A generic linear QTL model can then be formulated in matrix form,

$$\mathbf{y} = A(\mathbf{x})\mathbf{b} + \epsilon, \tag{1}$$

where the $n \times k$-matrix $A(\mathbf{x})$ is the design matrix and ϵ is the error vector. In the QTL mapping procedure, this model is evaluated for different combinations of locations $\mathbf{x}$ in search of the best model fit. This corresponds to solving the optimization problem

$$RSS_{opt} = \min_{\mathbf{b},\mathbf{x}} (A(\mathbf{x})\mathbf{b} - \mathbf{y})^T (A(\mathbf{x})\mathbf{b} - \mathbf{y}). \tag{2}$$

The problem (2) can be separated into two parts. The first, inner problem is to compute the model fit for a given position $\mathbf{x}$, referred to as the objective function evaluation (3). The second, outer problem is a global optimization search for this objective function, over the complete genome (4).

$$RSS(\mathbf{x}) = \min_{\mathbf{b}} (A(\mathbf{x})\mathbf{b} - \mathbf{y})^T (A(\mathbf{x})\mathbf{b} - \mathbf{y}), \tag{3}$$

$$RSS_{opt} = \min_{\mathbf{x}} RSS(\mathbf{x}), \tag{4}$$

The global search is a d-dimensional global optimization problem. When searching for d QTL, the search space is in principle a d-dimensional hypercube where the side is G. However, in practice, symmetries can be used to reduce the size of the search space somewhat. Methods for mapping of single QTL where the effects of other putative QTLs is taken into account have been derived [23, 24, 42]. When a QTL has been identified, it is also possible to include it in the model as a fixed effect, allowing for subsequent searches for additional QTL while accounting for this

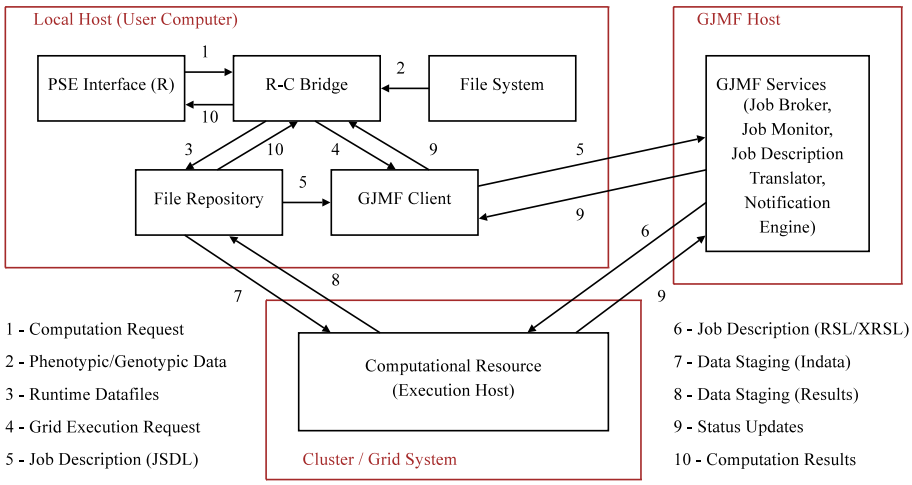

Figure 1: *The overall architecture*

effect [11]. This procedure is called *forward selection*, and drastically reduces the computational demand compared to a simultaneous search for multiple QTL. However, it has been shown that such methods can be ineffective in detecting interacting QTL [14].

In many QTL mapping codes, the search problem (4) is solved using a brute-force *exhaustive search*. In [26] we showed that it is possible to perform this type of search for multiple QTL on a grid system, where the search space is partitioned across many grid nodes. This reduces the time needed for the analysis, but the time requirement still becomes too demanding when the search dimensionality increases. For high-dimensional problems we would require an unrealistically large number of computational nodes for the results to be acquired within reasonable time limits.

DIRECT [29] (*DIviding RECTangles*) is a general global search algorithm, based on a process of partitioning the search into regions and then recursively refining the partitioning in those regions where a global minimum is most likely to be found, until a pre-determined stopping criterion is reached. A modified version of DIRECT [32] has been successfully used for the multidimensional QTL search problem (4). We have been able to extend this work in [25, 27], allowing the search to be done via the use of grid resources.

We have also previously demonstrated how the DIRECT-based QTL analysis software can be deployed and used via the Lunarc grid portal [18, 28]. In that work, the user was provided with a user-friendly interface for interacting with the grid resources. For a novice user, this portal-based system might be very advantageous, as there is no need for any specific software other than a web browser for performing simultaneous searches for multiple QTL and randomization tests. The QTLexpress service [36], its successor GridQTL [22] and the eQTL Infrastructure project [3] provide similar portal-based services.

Compared to a portal-based solution, our R-based PSE has the advantage that it can provide a much richer set of tools to the more advanced users. There are many third-party tools and packages which can be integrated into the R system, most of them available through the coordinated CRAN repository [5]. R comprises a large variety of statistical tools and graphics capabilities, and several application-specific packages for quantitative genetics have been released building upon this general functionality. R/qtl [12] is one such package which is widely used to perform different analysis in mainly inbred population structures. It provides various tools to import data files in several formats, and to perform QTL analysis using different parameters, as well as visualizing the results using different graphical methods. There are also graphical and text-based tools for QTL analysis tasks, like QTL Cartographer [10].

3 Architecture

A main requirement of the PSE architecture is that it should provide sufficient workflow functionality to be used by an experimentalist for controlling advanced QTL search tasks. A workflow system [17] typically manages the flow of data between different components of a system, where these components can be command-line applications or scripts which are self-contained. The communication between components can be controlled via standard streams or data files. Some workflow systems provide a GUI, where the data paths can be designed via drag-and-drop interfaces.

One common use of a workflow systems is to perform parameter sweeps. Here a given task is executed with different data or parameters. Within R, this type of functionality is easily achieved using a standard R loop. Listing 1 shows a script where a QTL search task is executed for stopping criteria with varying strictness. There is also an example of how this type of workflow can be used for randomization testing described in Section 4.

In this context, it is important to note that in an QTL analysis the correct parameter set which gives the best biologically significant results is normally not known. Here, our simple workflow system can be used as the basis for a model selection mechanism. By performing multiple QTL searches in a parameter sweep and then post-processing these results we can find the most significant results with an appropriate biological interpretation.

Listing 1: An Example Parameter Sweep in R

```
gbl<-DirectReadGlobal()
dta<-DirectReadData(gbl)
results<-vector(mode='list',length=10)
for(i in 1:10) {
    gbl$maxfval = i*1000
    result[[i]]<- DirectSearch(gbl,dta)}
```

One of the main requirements of our PSE architecture is that a rich set of data translation and visualization functions should be available. For example, functions for loading the necessary input data into R (data path 2 in figure 1) and for performing preliminary analysis to weed out genotyping errors or to select a subset of the total dataset can easily be be included in the framework. Here it should be noted that the representation of input data for QTL analysis has not been standardized, and thus there are many different formats used. Since R is a popular environment for data analysis, there is a large foundation of code in existence for reading data file formats into R. Our system can support these functions, with the addition of specific translation functions for data conversion for use within the PSE. R also has a rich set of visualization functions. Many QTL applications written for R have developed these further, and these routines can now be utilized by our PSE with very little added work.

Most of the components of our PSE have already been developed independently and have not been custom built for this system. The software has been developed in different programming languages. The QTL search code is coded in C/C++, some of the scripts for grid tasks are coded in Python and GJMF described below is Java-based. We use R scripts for interfacing all of these into one system, providing the end user with a consistent environment.

Another main requirement of the architecture is that of transparent execution of the QTL search, irrespective of the underlying computational hardware. If needed, the user can explicitly indicate if a search should be done on the grid, another external resource such as a cluster, or locally. The architecture also makes it possible to automatically let the PSE decide on this based on parameters such as the number of randomization tests, search dimensionality etc. The input data for the main QTL analysis code contain the phenotypic and genotypic values for the population as well as certain parametric data that describe how the QTL search should be performed. The main output generated is the location of the global minimum.

If a grid or another external, large-scale computational resource is used, the goal is to reduce the time for the analysis requested. When a grid-based QTL search is submitted, the search can be partitioned across many nodes. The independent tasks in a randomization test are also distributed so that different nodes do different randomization runs. In both cases, functions for post-processing of the different tasks and aggregation of the results are required. These tasks are done using external scripts which are coded using Python [38] and Perl [39]. These scripts parse the text file output from the analysis code and generate a single, serialized datafile which can be accessed from R. This approach allows a gradual integration of existing tools, rather than rewriting working and tested code.

3.1 Data Transfer

There are several data files that are used when performing a QTL search. To aviod the transfer of multiple files to the grid nodes, the interface code first constructs the data structures necessary for performing the search into a serialized data file, implemented using the Boost C++ library [1]. This serialized data file (data path 3 in figure 1) is then the only file needed by the code for execution on a remote node. However, the most obvious benefit of this approach is not a convenience just in data transfer, but rather also a way of making sure that that the full state of a job task is included in the serialized data. When a job is to be performed locally, the very same state data structures are prepared in the initialization code. The only difference is that instead of serializing the state, for transfer over the grid, a pointer to the state (job description) is just passed in a local function call, within the same operating system process. It is even possible to extend this approach into a more general "grid remoting" scheme where the aggregating, less computationally demanding, code is all written in C++, results and jobs being moved to grid nodes in a way that is transparent not only to the end user, but also to the application code developer.

It is preferred that the executable code to be submitted to the remote node is a static binary, which does not depend on specific libraries, as dynamic library dependencies can be problematic to satisfy on a heterogeneous set of grid nodes. Despite static linking, there is still a need to provide several compiled versions of the code, which can be uploaded to the remote node depending on its configuration. The nodes may have different CPU architectures and/or instruction sets. Proper selection of an executable binary reduces the overheads from data transfer, and also allows efficient use of different types of grid nodes. The *File Repository* in figure 1 can be located on a `gridftp` server, or an appropriate service can be enabled on the host machine.

3.2 Execution on the grid

Compared to performing tasks on a local host, grid jobs add overhead for file transfer and job management. However, as the computational cost of tasks increase, the parallelization offered by grids far outweigh this overhead.

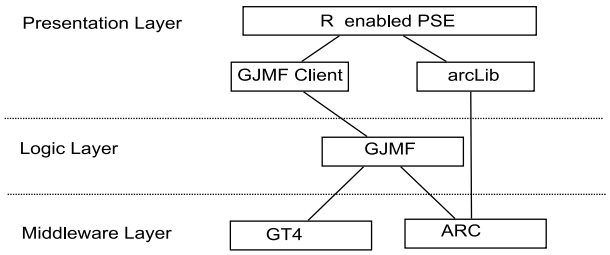

Figure 2: *The R - Grid Integration*

As seen from Table 1, an increase in search dimensionality increases execution time dramatically. Simple searches in 2 or 3 dimensions can be done on host machines, while searches of higher dimensionality are more efficiently performed distributed over grid resources. Distribution of searches onto grid resources also allows full availability of local systems while searches are in progress. Grid or batch execution of software can introduce certain constraints, e.g. the need for job description files. Job descriptions are used to automate execution of programs, and specify job execution parameters such as executable, runtime parameters, datafiles etc. As a number of different grid middlewares (that each define their own job description format) exist, an abstraction to make the majority of the PSE code independent of specific job description formats is needed. For this, and handling of other grid-specific issues such as data transfers, job monitoring etc, we employ the Grid Job Management Framework (GJMF) [16, 19].

The GJMF provides middleware-transparent access to grid functionality, allowing our PSE to offload grid-related issues to existing abstraction layers and avoid being limited to use of a specific grid middleware. In the case of job descriptions, our PSE generates a generic job description which the GJMF translates on demand to the format used by the target middleware when submitting a job. The GJMF also provides a reliability system, where the framework handles job failures and resubmission tasks. In large configurations, involving (tens of) thousands of nodes, even systems with very high levels of node and job reliability require this type of functionality to achieve reliable aggregated results. The GJMF provides a job status notification system which is used to determine when the search is completed.

The GJMF is implemented as a distributable set of web services, and does not need to reside on the same system as the grid resources, insulating the framework from grid hardware failures. The GJMF layer may, but does not have to, be hosted on the same machine as the R interface. A total of three different hardware systems can be involved in the job submission process, as described in Figure 2.

3.3 Computation of Genotype Probabilities

The genetic data is only known at the genetic marker locations specified by the experimental setup used for analyzing the genomes. IM (*Interval Mapping*) [31] methods determine the genotype probabilities at positions of incomplete marker information, and are also used to perform QTL analysis using limited marker data. Marker data can be limited due to missing observations, or ambiguities where the marker genotype does not indicate allele origin. The genotype probabilities then need to be inferred from surrounding markers. In our PSE, either marker data or a pre-computed dense mesh of genotype probabilities can be used. Most software and algorithms for these processes were developed when the number of markers was small. At present, the use of dense SNP (*single-nucleotide polymorphism*) marker maps make such methods infeasible, due to the change in marker count as well as the limited parsimony in individual markers in these maps, making the ambiguity problems mentioned above more frequent and apparent. The software system described in [34] uses a HMM (*Hidden Markov Model*) approach to more efficiently calculate the genotype probability estimates, even in outbred settings where marker ambiguity can be the most common state, rather than an exception. Since the DIRECT optimization algorithm used for the QTL search does not evaluate the objective function (3) at all loci, the HMM approach can be utilized to generate the genotypic probabilities on-the-fly at the actual positions needed rather than having them pre-computed. This will reduce the size of the input datafile for the QTL analysis jobs. This approach also reduces the computational load on the host, where the mesh would otherwise generally be calculated (if not, this work would be duplicated on all nodes). Finally, a proper application of the HMM approach leads to correct probabilities even when the putative QTL locations are closely linked and the probability for a compound genotype at the two loci is no longer decomposable into two independent events.

3.4 The QTL Search

Figure 3 describes the internal components of the *R-C interface* code in Figure 1. The codebase provides support for several global search methods, including a classical exhaustive search and the DIRECT algorithm. Other optimization methods, such as genetic algorithms [13] and hybrid global-local schemes [33] can be included to extend the existing prototype. Furthermore, different forward-selection strategies can easily be implemented in the R framework, to facilitate comparisons to existing work by others or to implement various suggested model selection strategies [11, 41].

The evaluation of the objective function (3) can be done in multiple ways. These include different formulations [8] of the linear regression (with additional variations possible based on covariates and what interaction parameters to include), but also variance component models [21]. Here, the computational load is several orders of magnitude higher compared to linear regression, despite different optimizations and approximations. While variance component methods pose different demands on input data, they can still be implemented in our overall framework, where the optimization strategies would be unchanged.

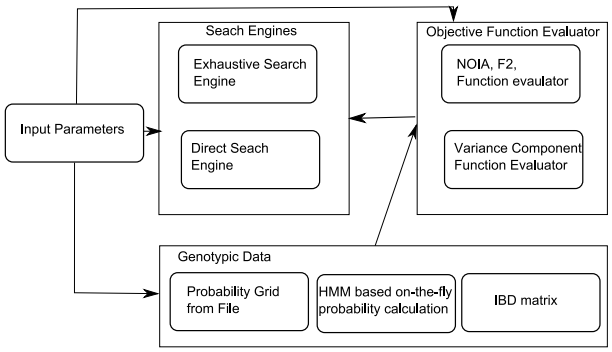

Figure 3: *Internal architecture of the QTL search code*

4 Proof-of-Concept Implementation

We have implemented a proof-of-concept version of the PSE described in general terms in the previous section. This implementation is aimed at looking into the different issues that may need to be addressed in a full implementation of the system. The modularity in the design means that most issues that can arise will be found by just implementing one or two realizations of each component type and deciding on preliminary interfaces between these. Furthermore, many of the practical challenges in interfacing to the complex ecosystem of surrounding software (grid middleware, the R environment, GJMF, Python, Perl) also surface early in the process and have been handled during the implementation of the initial version.

The QTL search code was previously used as a stand-alone application. The data and search parameters were passed using a text file, and a list of command-line parameters. This file has been replaced by an XML file which is more user-friendly when command-line use is still desired. This file is also used by the prototype for reading default global parameters, such as the search parameters, the default path for data files, etc. If the user does not specify any details, these parameters are thus meant to be read using the XML file. From within R, the user can explicitly specify a separate file, or specify these parameters manually.

Among the internal components described in Figure 3 we have implemented the DIRECT search engine. There is support for reading the genotypic data as a probability grid file as well as the HMM based methods described in [34]. The support for variance component based methods has not been implemented in the prototype.

As the standalone version of the code already has functions to read the data containing individual genotypes and phenotypes, these functions are now called from within R to populate the data in R data structures. Since there are many different data file formats used with QTL analysis, some of them unique to different tools such as QTLexpress [36], R/qtl [12], another usecase is to call R functions already developed for reading these data files. Then a simple translator can convert the data into the format used by our code.

Once the data have been read into R, we can use the functionality provided within the environment to analyze and visualize them using builtin graphics capabilities or more advanced tools developed by others. We pass these data, and the search parameters, to the QTL search code. The function call to the search procedure can specify if the requested search should be done on the grid. The call is 'blocking' at present, and returns the global minimum located. This simplifies the process of automating a workflow including demanding grid-deployed parts as well as further local analysis steps, as the user writing automated scripts can simply include the QTL search function call as a normal command that returns when finished, despite the underlying loosely connected nature of the grid.

If we specify that the search is done externally, as a grid job, then the task is partitioned across several nodes and submitted. If the number of nodes is not specified, a default value is assumed. Individual grid jobs can return their results as structured text files. This is appropriate, as redirection of the process standard output is a common property of middleware solutions and the total amount of output is limited. This file is read into R with our supporting scripts allowing for analysis of the results. If further aggregation of results would be needed, the results could be deserialized back into the C++ code, creating the logical impression that all analysis was performed locally.

Since the results from one call can be used as the input to another call, within R we can easily perform randomization testing without the need for a different QTL analysis package specifically developed for this purpose. Listing 2 gives an example R user script performing this, in a homogeneous population structure.

Listing 2: Randomization testing using R

```
library("gtools")
gbl<-DirectReadGlobal()
dta<-DirectReadData(gbl)
random<-vector(mode='list',length=100)
for(i in 1:100){
    dta$yvec<-permute(dta$yvec)
    random[[i]]<- DirectSearch(gbl,dta)}
```

4.1 Re-analysis of experimental data

In [30], an F_2 chicken intercross was described between White Leghorn and Red Junglefowl, which was assessed for several growth-related phenotypes, including body weight at 1, 8, 46, 112, and 200 days after hatching. Multiple QTL were found, including some interactions [15].

As a test of the power of our PSE, we re-analyzed these data with an updated marker map (L. Andersson, pers. comm.), with probabilities computed using the methodology described in [34]. A subset of the national grid resource Swegrid [6] was used for the experiments, and analysis was performed for body weight at 1, 8, 46,

112, and 200 days after hatching, with models ranging from 1 to 4 simultaneously included loci. Furthermore, significance thresholds were derived through permutation testing for each of these, based on 1,000 permutations. Time gains thanks to the use of multiple grid nodes are presented in Table 2, based on the trait of 200 days after hatching. The number of nodes was chosen to illustrate the case of an end user with access to a grid or cluster environment with a rather limited allotment, possibly using otherwise unutilized "spare" resources.

We focus here on the two traits with the highest heritability according to the existing results, 112 and 200 days. For these two traits, all models were more than 99.9 % significant relative to the null hypothesis, due to a single very strong QTL. Based on inferring extreme-value distributions [35], we can choose the model with the highest theoretical rank in a hypothetical randomization run with a higher number of permutations, so compensating for the differences in dimensionality and the relative overfitting arising from the higher number of degrees of freedom.

With this method, the analysis for body weight after 112 days gives preference for a model of two loci on chromosome 1, at 104 and 484 cM, accounting for about 12.3% of the total phenotypic variance. These loci were also found in previous analysis, when compensating for location differences in the two maps. For body weight after 200 days, a 3-locus model was automatically selected, adding a locus on chromosome 2 at 193 cM, accounting for about 12.4%. Note that the total assessed genetic variance was not identical for these two phenotypes. The same two-locus model as chosen for the 112-day trait would only account for 11.9% at 200 days.

Table 2: *Randomization testing locally and on the grid.*

Dimension	Local runtime(s)	Grid runtime(s)	Grid Jobs (Speedup)
1	72	13	10 (6)
2	1765	353	10 (5)
3	26613	1558	20 (17)
4	352789	2565	207 (138)

5 Conclusions

We have presented a PSE that enables biologists within quantitative genetics to exploit already existing software components within a grid framework. In practical use, the complexities related to performing the QTL search task in a grid are made invisible to the user. A user can also perform simpler computations locally on a standard laptop using the same PSE, requiring no change in environment. The inherent features of R, combined with the significant community support, makes this statistical computing environment an ideal grid-enabled workflow system for QTL analysis, with the possible exception of extremely large datasets where the included components can instead be called directly. Our PSE allows the biologist to perform necessary computational and data translation services with a high degree of flexibility, efficiency, and performance.

The proof-of-concept implementation has illustrated that this type of system can be implemented with relative flexibility using existing components. The PSE integrated into R can be further enhanced with existing codes and tools, given the proper interfaces and data translation services.

We have demonstrated how to use the R programming environment to perform randomization testing, and other data analysis tasks can also be implemented in just a few lines of R code. Our experimental results show that the time savings in a practical analysis are considerable for models where 3 or 4 loci are searched for, especially when considering that an actual experimental workflow might include different sets of covariates, variations in model structure, transformations of phenotype values to approximate different statistical distributions, etc., all of these while allowing a higher number of simultaneous loci than 2.

In the future, we hope to optimize the randomization testing further, as some search tasks can be stopped without the need to wait for completion, thanks to the mathematical properties of the objective function. We will also improve the user scenario of interacting with other popular R-enabled QTL analysis software. Some of these packages provide superior services for pre-processing and visualization of results, while we can then focus on the genotype probability model and the multidimensional global searches, both of which are computationally demanding and require the kind of HPC approach that makes our package unique among systems that are accessed programmatically (rather than through a web portal). This is also evidenced by the ease with which we could perform the re-analysis of experimental data.

We also hope to look into the integration of variance component models within the search framework, as efficient implementations of these very computationally demanding models, in a multi-dimensional global search setting, would be an ideal task for grid distribution. However, even for linear-regression models, the kind of multidimensional analyses needed to properly identify loci with epistatic effects in real datasets [14, 15] can take days for more than two interacting loci on a single machine. Our approach allows repeated such analyses, complete with visualization, data aggregation and permutation-based significance testing, to be done with a turnaround time of less than an hour, thus opening for novel scientific workflows during exploratory analysis.

6 Acknowledgements

The work by C.N. has been funded by The Graduate School in Mathematics and Computing (FMB), Uppsala University, Sweden. Per Jensen and Leif Andersson are acknowledged for sharing experimental data for evaluation of the method.

References

[1] Boost C++ Libraries. http://www.boost.org.

[2] EMBRACE. http://www.embracegrid.info.

[3] eQTL Project. http://eqtl.berlios.de.

[4] Lunarc application portal toolkit. www.lunarc.lu.se/Software/lap/.

[5] The R project for statistical computing. http://www.r-project.org.

[6] Swegrid. http://www.swegrid.se.

[7] TERAGRID. http://www.teragrid.org/gateways.

[8] J. M. Álvarez-Castro and Ö. Carlborg. A unified model for functional and statistical epistasis and its application in quantitative trait loci analysis. *Genetics*, 176(2):1151–1167, 2007.

[9] T.E. Athanaileas, D.I. Kaklamani, and G.V. Tsoulos. A grid enabled problem solving environment for monte carlo matlab simulations. In *DS-RT '07: Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 159–166, Washington, DC, USA, 2007. IEEE Computer Society.

[10] C. Basten, B. Weir, and Z.-B. Zeng. *QTL Cartographer, version 1.15*. Dept. of Statistics, North Carolina State University, Raleigh, NC, 2001.

[11] Karl W. Broman. *Identifying quantitative trait loci in experimental crosses*. PhD thesis, Department of Statistics, University of California, Berkeley, 1997.

[12] K.W. Broman, H. Wu, S. Sen, and G.A. Churchill. R/qtl: QTL mapping in experimental crosses. *Bioinformatics*, 19(7):889–890, 2003.

[13] Ö. Carlborg, L. Andersson, and B. Kinghorn. The use of a genetic algorithm for simultaneous mapping of multiple interacting quantitative trait loci. *Genetics*, 155:2003–2010, 2000.

[14] Ö. Carlborg and C.S. Haley. Epistasis: too often neglected in complex trait studies? *Nature Reviews Genetics*, 5:618–625, 2004.

[15] Ö. Carlborg, S. Kerje, K. Schtz, L. Jacobsson, P. Jensen, and L. Andersson. A global search reveals epistatic interactions between QTL for early growth in chicken. *Genome Res.*, 13:413–421, 2003.

[16] E. Elmroth, P. Gardfjäll, A. Norberg, J. Tordsson, and P-O. Östberg. Designing general, composable, and middleware-independent grid infrastructure tools for multi-tiered job management. In T. Priol and M. Vaneschi, editors, *Towards Next Generation Grids*, pages 175–184. Springer-Verlag, 2007.

[17] E. Elmroth, F. Hernández, and J. Tordsson. Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. *Future Generation Computer Systems. The International Journal of Grid Computing: Theory, Methods and Applications*, 26(2):245–256, February 2010.

[18] E. Elmroth, S. Holmgren, J. Lindemann, S. Toor, and P-O. Östberg. Empowering a flexible application portal with a SOA-based grid job management framework. Accepted for publication in Proc. PARA8, May 13-16, Trondheim, Norway, 2008.

[19] E. Elmroth and P-O. Östberg. GJMF - a service-oriented grid job management framework. Submitted for journal publication, 2009.

[20] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufman, 1999.

[21] D.A. Harville. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*, 72(358):320–338, 1977.

[22] J. Hernandez-Sanchez, J-A. Grunchec, and S. Knott. A web application to perform linkage disequilibrium and linkage analyses on a computational grid. *Bioinformatics*, 25(11):1377–1383, June 2009.

[23] R. Jansen. A general mixture model for mapping quantitative trait loci by using molecular markers. *Theoretical And Applied Genetics*, 85:252–260, 1992.

[24] R. Jansen. Interval mapping of multiple quantitative trait loci. *Genetics*, 135:205–211, 1993.

[25] M. Jayawardena and S. Holmgren. Grid-enabling an efficient algorithm for demanding global optimization problems in genetic analysis. In *3rd IEEE International Conference on e-Science and Grid Computing, IEEE Conference Proceedings 10.1109*, pages 205–212. IEEE, 2007.

[26] M. Jayawardena, S. Holmgren, and K. Ljungberg. Using parallel computing and grid systems for genetic mapping of multifactorial traits. In B. Kågström, E. Elmroth, J. Dongarra, and J. Waśniewski, editors, *Applied Parallel Computing: State of the Art in Scientific Computing, Lecture Notes in Comp. Sci*, pages 627–636. Springer-Verlag, 2007.

[27] M. Jayawardena, S. Holmgren, and H. Löf. Efficient optimization algorithms and implementations for genetic analysis of complex traits on a grid system with multicore nodes. Accepted for publication in Proc. PARA08, May 13-16, Trondheim, Norway, 2008.

[28] M. Jayawardena, S. Toor, and S. Holmgren. A grid portal for genetic analysis of complex traits. In *MIPRO 2009 International Conference on Grid and Visualisation systems, Opatija, Croatia, May 25-29*, 2009.

[29] D. Jones, C. Perttunen, and B. Stuckman. Lipschitzian optimization without the lipschitz constant. *J. Optimization Theory App*, 79:157–181, 1993.

[30] S. Kerje, Ö. Carlborg, K. Schütz, C. Hartmann, P. Jensen, and L. Andersson. The twofold difference in adult size between the red junglefowl and white leghorn chickens is largely explained by a limited number of QTLs. *Animal Genetics*, 34(4):264–274, 2003.

[31] E. Lander and D. Botstein. Mapping mendelian factors underlying quantitative traits using RFLP linkage maps. *Genetics*, 121:185–199, 1989.

[32] K. Ljungberg, S. Holmgren, and Ö. Carlborg. Simultaneous search for multiple QTL using the global optimization algorithm DIRECT. *Bioinformatics*, 20:1887–1895, 2004.

[33] K. Ljungberg, K. Mishchenko, and S. Holmgren. Efficient algorithms for multi-dimensional global optimization in genetic mapping of complex traits. Technical report 2005-035, Division of Scientific Computing,Dept of IT, Uppsala University, 2005.

[34] C. Nettelblad, S. Holmgren, L. Crooks, and Ö. Carlborg. cnf2freq: Efficient determination of genotype and haplotype probabilities in outbred populations using markov models. In Sanguthevar Rajasekaran, editor, *BICoB 2008*, volume 5462 of *Lecture Notes in Computer Science*, pages 307–319. Springer, 2009.

[35] L. Rönnegård, F. Besnier, and Ö. Carlborg. An improved method for quantitative trait loci detection of within-line segregation in F2 intercross designs. *Genetics*, 2008.

[36] G. Seaton, C. Haley, S. Knott, M. Kearsey, and P. Visscher. QTL express: mapping quantitative trait loci in simple and complex pedigrees. *Bioinformatics*, 18:339–340, 2002.

[37] Mary Thomas, Maytal Dahan, Kurt Mueller, Stephen Mock, Catherine Mills, and Ray Regno. Application portals: practice and experience. *Concurrency and Computation: Practice and Experience*, 14(13-15):1427–1443, 2002.

[38] Guido Van Rossum. *The Python language reference manual*. Network Theory Ltd., September 2003.

[39] Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl (3rd Edition)*. O'Reilly, July 2000.

[40] A. YarKhan, J. Dongarra, and K. Seymour. Gridsolve: The evolution of network enabled solver. In P. Gaffney and C.T. James, editors, *Grid-Based Problem Solving Environments: IFIP TC2/WG 2.5 Working Conference on Grid-Based Problem Solving Environments*, pages 215–226. Springer, 2007. (Prescott, AZ, July 2006).

[41] Malgorzata Zak, Andreas Baierl, Malgorzata Bogdan, and Andreas Futschik. Locating multiple interacting quantitative trait loci using rank-based model selection. *Genetics*, 176(3):1845–1854, 2007.

[42] Z.-B. Zeng. Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. *Proc Nat Acad Sci USA*, 90:10972–10976, 1993.