DAIRRy-BLUP: A high performance computing approach to genomic prediction

A. De Coninck*, J. Fostier†, S. Maenhout‡, B. De Baets*

* KERMIT, Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, B-9000 Ghent, Belgium; † IBCN, Department of Information Technology, Ghent University - iMinds, B-9000 Ghent, Belgium; ‡ Progeno, B-9052 Zwijnaarde, Belgium

ABSTRACT

In genomic prediction, common analysis methods rely on a linear mixed model framework to estimate SNP marker effects and breeding values of animals or plants. Ridge regression–best linear unbiased prediction (RR-BLUP) is based on the assumptions that SNP marker effects are normally distributed, uncorrelated and have equal variances. We propose DAIRRy-BLUP, a parallel, distributed-memory RR-BLUP implementation that uses the average information algorithm for restricted maximum likelihood estimation of the variance components. The goal of DAIRRy-BLUP is to enable the analysis of large-scale datasets in order to provide more accurate estimates of marker effects and breeding values. A distributed-memory framework is required since the dimensionality of the problem, determined by the number of SNP markers, can become too large to be analyzed by a single computing node. Initial results show that DAIRRy-BLUP enables the analysis of very large-scale datasets (up to 1,000,000 individuals and 360,000 SNPs) and indicate that increasing the number of phenotypic and genotypic records has a more significant effect on the prediction accuracy than increasing the density of SNP arrays.

Copyright 2014.
INTRODUCTION

In the field of genomic prediction, genotypes of animals or plants are used to predict either phenotypic properties of new crosses or breeding values (EBVs) for detecting superior parents. Since quantitative traits of importance to breeders are mostly regulated by a large number of loci (QTL)\(^2\), high density SNP markers are used to genotype individuals. The most frequently used SNP arrays for cattle consist of 50,000 SNP markers, but even genotypes with 700,000 SNPs are already available (Cole et al. 2012).

Some widely-used analysis methods rely on a linear mixed model backbone (Meuwissen et al. 2001) in which the SNP marker effects are modelled as random effects, drawn from a normal distribution. The predictions of the marker effects are known as BLUP, which are linear functions of the response variates. It has been shown that when no major genes contribute to the trait, Bayesian predictions and BLUP result in approximately the same prediction accuracy for the EBVs (Hayes et al. 2009; Legarra et al. 2011; Daetwyler et al. 2013).

At present the number of individuals included in the genomic prediction setting is still an order of magnitude smaller than the number of genetic markers on widely-used SNP arrays, favoring algorithms that directly estimate EBVs, which is in this case computationally more efficient than first estimating the marker effects (VanRaden 2008; Misztal et al. 2009; Piepho 2009; Shen et al. 2013). Nonetheless, it has been shown theoretically (Hayes et al. 2009) that in order to increase the prediction accuracy of the EBVs for traits with a low heritability, the number of genotyped records should increase dramatically. Most widely used implementations like synbreed (Wimmer et al. 2012) and BLUPF90 (Misztal et al. 2002) are limited by the computational resources that can be provided by a single workstation. We present DAIRRy-BLUP, a parallel framework that takes advantage of a distributed-memory compute cluster in order to enable the analysis of large-scale datasets. Additionally, results on simulated data illustrate that the use of such large-scale datasets is warranted as it significantly improves the prediction accuracy of EBVs and marker effects.

DAIRRy-BLUP is optimized for processing large-scale dense datasets with SNP marker data for a large number of individuals (>100,000). Variance components are estimated based on the entire dataset using the average information restricted maximum likelihood (AI-REML) algorithm. Therefore, it distinguishes itself from other implementations of the AI-REML algorithm, which are mostly optimized for sparse settings and are not able to make use of the aggregated compute power of a supercomputing cluster (Masuda et al. 2013; Misztal et al. 2002).

All matrix equations in the algorithms used by DAIRRy-BLUP are solved using a direct Cholesky decomposition. Although efficient iterative techniques exist that can reduce memory requirements and calculation time for solving a matrix equation (Legarra and Misztal 2008), the choice for a direct solver is motivated by the use of the AI-REML algorithm for variance component estimation on the entire dataset.

MATERIALS AND METHODS

Simulated data

The simulated data used for benchmarking DAIRRy-BLUP was generated using AlphaDrop (Hickey and Gorjanc 2012), which was put forward by the Genetics Society of America (GSA) to provide common datasets for researchers to benchmark their methods (de Koning and McIntyre 2012). Since no datasets with

\(^2\) E.g. more than 2000 QTLs are already found to contribute to the composition and production of milk in cattle (http://www.animalgenome.org/cgi-bin/QTLdb/BT/index)
more than 10,000 phenotypic and genotypic records are yet publicly available, this software was used to create datasets with 10,000, 100,000 and 1,000,000 individuals of a livestock population genotyped for a varying number of SNP sites (9,000, 60,000 and 360,000). AlphaDrop first launches MaCS (Chen et al. 2009) to create 4,000 haplotypes for each of the 30 chromosomes using an effective population size of 1,000 for the final historical generation, which is then used as the base generation for the next 10 generations under study. The base individuals of a simulated pedigree had their gametes randomly sampled from the 4,000 haplotypes per chromosome of the sequence simulation. The pedigree comprised 10 generations with 10 dams per sire and 2 offspring per dam, with the number of sires depending on the total number of records that was aimed for (50, 500 or 5,000). Genotypes for individuals from subsequent generations were generated through Mendelian inheritance with a recombination probability of 1% per cM. The phenotypes of the individuals were affected by 9,000 QTL, randomly selected out of the approximately 1,670,000 available segregating sites. The effects of these QTL were sampled from a normal distribution with mean 0 and standard deviation 1. Final phenotypes based on the QTL effects were generated with a user-defined heritability of 0.25.

When constructing the large-scale datasets, it became clear that AlphaDrop, which is a sequential program and does not make use of any parallel programming paradigm, was not designed for generating SNP data for such large numbers of individuals. The largest dataset that could be created consisted of 1,000,000 individuals, each genotyped for 60,000 SNPs. This required about 118 GB of working memory and the final data set was stored in text-files with a total size of about 350 GB. The simulation took more than 13 days on an Intel Xeon E5-2420 1.90 GHz with 64 GB RAM and another 65 GB as swap memory. The datasets can be recreated by using the correct seeds for the random number generators in the MaCS and AlphaDrop software.

**Statistical method: linear mixed model**

The underlying framework for the analysis of genotypic datasets is the linear mixed model:

\[ y = Xb + Zu + e \]

where \( y \) is a vector of \( n \) observations, \( b \) is a vector of \( t \) fixed effects, \( u \) is a vector of \( s \) random effects and \( e \) is the residual error. \( X \) and \( Z \) are the incidence matrices that couple the effects to the observations. Assuming that the random effects and the residual error are normally distributed: \( u \sim \mathcal{N}(0, G) \) and \( e \sim \mathcal{N}(0, R) \), the observations are also normally distributed: \( y \sim \mathcal{N}(Xb, V) \), with \( V = R + ZGZ' \).

The BLUP of the random effects are linear estimates that minimize the mean squared error and exhibit no bias. Henderson (1963) translated this estimation procedure into the so-called mixed-model equations (MME), which can be written in matrix form as:

\[
\begin{bmatrix} X'R^{-1}X & X'R^{-1}Z \\ Z'R^{-1}X & Z'R^{-1}Z + G^{-1} \end{bmatrix} \begin{bmatrix} b \\ u \end{bmatrix} = \begin{bmatrix} X'R^{-1}y \\ Z'R^{-1}y \end{bmatrix}
\]

These equations were conceived at a time when SNP markers were not yet available and the random effects were associated with a genotypic value per individual. However, when SNP markers were introduced in animal breeding, these equations were used with the SNP markers modeled explicitly and the genotypic value corresponding to the sum of the SNP marker effects (Meuwissen et al. 2001). In this study, SNP marker effects are always modeled explicitly to obtain a matrix equation whose dimensionality depends only on the number of SNP markers and the number of fixed effects included in the analysis. As such it is anticipated that genomic
datasets will mainly grow in the number of individuals due to the decreasing cost for genotyping (Wetterstrand 2014).

A common simplification, which leads to the ridge regression (RR) formulation (Piepho 2009), is to assume that all SNP marker effects are uncorrelated and have homoscedastic variance $\sigma^2_u$. Similarly, residual errors are assumed to be uncorrelated and to have homoscedastic variance $\sigma^2_e$. Although these assumptions might not be in accordance with reality, it has been shown that this simplified model can reach prediction accuracies that are close to those of more complex models (Shen et al. 2013; VanRaden et al. 2008). The covariance matrices are thus set to $\Sigma_u = \sigma^2_u I_n$ (with $I_n$ the identity matrix of dimension $n$) and $\Sigma_e = \sigma^2_e I_n$, which leads to the following MME:

$$
\begin{bmatrix}
X^T X & X^T Z \\
Z^T X & Z^T Z + \sigma^2_u I_n
\end{bmatrix}
\begin{bmatrix} b \\ u \end{bmatrix} =
\begin{bmatrix} X^T y \\ Z^T y \end{bmatrix}.
$$

The solution to this matrix equation is known as the Ridge Regression BLUP (RR-BLUP), because $\sigma^2_e / \sigma^2_u$ can be seen as a ridge regression penalization parameter, which controls the penalization by the sum of squares of the marker effects. In this case, however, the penalization parameter is not estimated by cross-validation, but since it is a ratio of two variance components, the latter will be estimated by a Restricted Maximum Likelihood procedure (REML).

In this specific study on simulated datasets, no fixed effects other than the overall mean are modelled, i.e. $t = 1$ and $X = 1_n$ (vector of $n$ ones). The random effects are the SNP marker effects and every observation corresponds to the phenotypic score of an individual.

**Average information REML (AI-REML)**

When the covariance structures of the random effects and of the residual errors depend on the respective variance component vectors $\gamma$ and $\phi$,

$$
\begin{bmatrix} u \\ e \end{bmatrix} \sim \mathcal{N}_p \left( 0, \sigma^2 \begin{bmatrix} G(\gamma) & 0 \\
0 & R(\phi) \end{bmatrix} \right),
$$

the REML log-likelihood can be written as follows (Gilmour et al. 1995):

$$
l_{\text{REML}}(\sigma^2, \gamma, \phi) = -\frac{1}{2} \left( (n-t) \log \sigma^2_e + \log |G| 
+ \log |R| + \log |C| + \frac{y^T Py}{\sigma^2_e} \right),
$$

where $C$ is the coefficient matrix from the MME as defined in the previous section and

$$
P = V^{-1} - V^{-1} X (X^T V^{-1} X)^{-1} X^T V^{-1}.
$$

With the aforementioned assumptions and setting $\sigma^2 = \sigma^2_e$, $R = I_n$, and $G = \gamma I_n$, with $\gamma = \sigma^2_u / \sigma^2_e$, this simplifies to a log-likelihood with only 2 parameters, namely $\sigma^2_e$ and $\gamma$:

$$
l_{\text{REML}}(\sigma^2_e, \gamma) = -\frac{1}{2} \left( (n-t) \log \sigma^2_e + b \log \gamma 
+ \log |C| + \frac{y^T Py}{\sigma^2_e} \right),
$$

where $C$ and $P$ only depend on $\gamma$. In this form, we can find an analytical solution for the maximization of the log-likelihood with respect to $\sigma^2_e$, provided $\gamma$ is known:

$$
\sigma^2_e = \frac{y^T Py}{n-t}.
$$

The maximization of the REML log-likelihood function with respect to $\gamma$ can mostly not be solved analytically, due to
the complex first derivative with respect to \( \gamma \), also known as the score function:

\[
\frac{\partial l_{\text{REML}}}{\partial \gamma} = -\frac{1}{2} \left( b - \text{tr}(C_{\text{ZZ}}) - \frac{\hat{u}'\hat{u}}{\hat{\sigma}^2\gamma^2} \right),
\]

(1)

with \( C_{\text{ZZ}} \), the lower right block of the inverse of \( C \), also depending on \( \gamma \). Newton's method for maximizing likelihood functions can be used to find a solution for this optimization problem. The iterative scheme looks like:

\[
\kappa_{i+1} = \kappa_i - H_i^{-1}\nabla l_{\text{REML}}(\kappa_i),
\]

(2)

with \( \kappa_i = (\hat{\sigma}^2, \gamma_i) \) at iteration \( i \), \( H_i \) the Hessian matrix of \( l_{\text{REML}} \) evaluated at \( \kappa_i \) and \( \nabla l_{\text{REML}}(\kappa_i) \) is the gradient vector of the REML log likelihood with respect to \( \kappa \), evaluated at \( \kappa_i \). However, the Hessian matrix can be very hard to construct due to the large size of the matrices that need to be multiplied or inverted.

Originally, Patterson and Thompson (1971) used the Fisher information matrix, which is the expected value of the Hessian matrix, for updating the variance components, but the computation of the elements of this Fisher information matrix involves calculating traces of large matrices, which are tedious to construct. Taking an average of the expected and observed Hessian matrix, these traces disappear in the expression of the update matrix. The resulting update matrix is called the Average Information matrix (AI-matrix) and it can be shown (Gilmour et al. 1995) that this AI-matrix can easily be calculated as:

\[
H_{\text{AI}} = \frac{1}{2\hat{\sigma}^2}\mathbf{Q}'\mathbf{PQ},
\]

\[
\mathbf{Q} = \begin{bmatrix}
\mathbf{y} & \mathbf{Z}\hat{u} / \gamma
\end{bmatrix},
\]

(3)

A more detailed derivation of the AI-matrix can be found in the appendix.

Not only is the AI-REML procedure computationally practical for use in a distributed-memory framework, it has also been shown that less iterations are needed to obtain convergence compared to a derivative-free (DF) algorithm or an expectation-maximization (EM) algorithm (Gilmour et al. 1995). Despite the fact that the computational burden of the AI algorithm is somewhat heavier compared to both DF and EM, the total computation time might be significantly lower due to rapid convergence.

**Distributed-memory computing techniques**

For large-scale datasets, the application of the RR-BLUP methodology on a single workstation is computationally impractical because of the high memory requirements to store the large and dense coefficient matrix \( C \). In order to deal with datasets beyond the memory capacity of a single machine, a parallel, distributed-memory implementation was developed. DAIRRy-BLUP can take advantage of the aggregated compute power and memory capacity of a large number of networked machines by distributing both data and computations over different processes, which can be executed in parallel on the CPU cores of the networked nodes. This allows DAIRRy-BLUP to be executed even on clusters with a low amount of memory per node, because every large matrix in the algorithm is distributed among all parallel processes, i.e. every process holds parts of the matrix and can pass these parts to another process if necessary through message passing (Snir et al. 1995). By default the matrix is split up in blocks of 64 by 64 elements, however, this can be changed in function of matrix size and computer architecture. More information about how matrices are distributed in DAIRRy-BLUP can be found in the appendix.
The coefficient matrix $C$ has dimensions of $(s + t) \times (s + t)$, the sum of random and fixed effects, however, in order to construct it, we need to multiply matrices with dimensions $s$, the number of random effects, by $n$, the number of individuals, and $t$, the number of fixed effects, by $n$. In genomic datasets, $s$, the number of SNP markers, is usually a lot larger than $t$ and nowadays $n$ is mostly still smaller than $s$. However, as will be discussed in further sections, the number of observations ($n$) should increase to provide accurate estimates of the marker effects and the resulting breeding values. It is expected that indeed the number of genotyped individuals will rise in the future, since the cost of genotyping is constantly decreasing and more data of genotyped animals will be gathered in future years. Therefore, the algorithm is implemented in such a way that the number of observations $n$ does not have an impact on memory usage. To obtain this, the $y$-vector and the $Z$ and $X$ matrices are read from the input files per strip, as was also suggested by Piepho et al. (2012). These strips consist of a certain number of rows, defined by the block size and the number of processes involved. The lower-right block of the coefficient matrix $C_{zz}$ is then calculated as follows:

$$C_{zz} = \sum_{i=1}^{q} Z_i^T Z_i + \gamma I_Z,$$

(4)

with $q$ the number of strips needed to read in the entire $Z$ matrix, $Z_i$ the strips of the $Z$-matrix and $I_Z$ the identity matrix with the same dimensions as $C_{zz}$.

All mathematical operations that have to be performed on the distributed matrices rely on the standard distributed linear algebra libraries PBLAS (Choi et al. 1996) and ScaLAPACK (Blackford et al. 1997). The MME are solved through a Cholesky decomposition of the symmetric coefficient matrix $C$. The AI-REML algorithm iterates until convergence of $\gamma$ and since predictions of the marker effects, depending on $\gamma$, appear in the score function of the AI-REML algorithm, a Cholesky decomposition of the updated coefficient matrix is required each iteration. ScaLAPACK overwrites the original coefficient matrix by its Cholesky decomposition, implying $C$ to be set up from scratch at the beginning of every iteration. Nonetheless, when sufficient working memory is available and the construction of $C$ becomes a time-consuming factor, an option can be enabled in DAIRRy-BLUP to store a copy of the coefficient matrix, which can then rapidly be updated with new values of $\gamma$ and thus renders a complete set-up of $C$ unnecessary.

**Input files**

As already mentioned, the data files can require more than 100 GB of disk space when stored as a text file. Due to the distributed read-in of the data files, DAIRRy-BLUP can only cope with binary files, which may require even more space when the data is stored as 64 bit integers or doubles. To overcome this massive hard disk consumption, DAIRRy-BLUP can read in data in the Hierarchical Data Format (HDF5) (The HDF Group 2000-2010). The HDF5 file format stores datasets in a filesystem like format, providing every dataset with a `/path/to/resource` path. Metadata can be added to the datasets in the form of named attributes attached to the path or the dataset. It is recommended to use the HDF5-filetype for analyzing data with DAIRRy-BLUP due to 2 major advantages.

First of all, HDF5 offers the possibility to compress data into blocks such that up to 10 times less space is consumed by the final HDF5 file than by storing the data in text files. Secondly, if the size of the compressed blocks is equal to the size of the blocks in which the data is distributed across the memory of all dedicated processes, there is also a significant gain in read-in time of the data. Indeed, every process only needs to decompress several
blocks before reading in and doesn't have to go through the whole dataset as is the case when using conventional binary files.

**High performance computing infrastructure**

All results were obtained using the Gengar cluster on Stevin, the high performance computing (HPC) infrastructure of Ghent University. This cluster consists of 194 computing nodes (IBM HS 21 XM blades) interconnected with a 4X DDR Infiniband network (20 Gbit/s). Each node contains a dual socket quad-core Intel Xeon L5420 2.5 GHz CPU (8 cores) with 16 GB RAM. To achieve a high performance a one-to-one mapping of processes to CPU cores was applied and the number of dedicated CPU cores was chosen in such a way that there was a fair balance between wall time and occupation of the Gengar cluster, ensuring that the limit of two GB RAM per core was not exceeded.

**RESULTS**

The results illustrate the capability of DAIRRy-BLUP to analyze large-scale datasets in a reasonable amount of time, when sufficient computing power is available. Table 1 summarizes the computational demands for the different datasets with 100,000 genotyped individuals. It is clear from these results that it becomes computationally impractical to use only a single computing node due to the high amount of RAM required. Implementations that use a genomic relationship matrix (VanRaden 2008) to estimate genomic EBVs directly, will in this case require at least 40 GB of RAM when only the upper or lower triangular of the symmetric coefficient matrix is stored. There are machines available with this amount of memory, however, the distributed-memory implementation has the advantage of being able to use as much working memory as available on all networked computers, and computing power can thus easily be increased by adding more machines to the network.

Another advantage of DAIRRy-BLUP is the fact that memory usage depends only on the number of fixed and random effects, and not on the number of individuals included in the analysis. Adding information on more individuals only influences the time required for reading the data from the input files and constructing the coefficient matrix, which is observed to scale linearly with the number of individuals. For datasets where genotypic information is obtained with a certain SNP chip, the required amount of working memory will remain constant regardless of the number of individuals included in the analysis. As will be shown further on, prediction accuracy of EBVs increases more significantly when adding more individuals to the analysis than when using denser SNP arrays, at least when the SNP arrays are already sufficiently dense so that the QTL and SNP markers are in linkage disequilibrium (Habier et al. 2013). DAIRRy-BLUP can easily be used on the same machine for such datasets, where the number of genotyped individuals is constantly increasing. Therefore, more accurate estimates of EBVs are obtained with minimal effort.

**Estimated versus True Breeding Values**

Since breeding values are of utmost importance to breeders, it is of course essential to be able to predict these breeding values correctly. In the datasets, the traits were defined by 9,000 random QTL, whose effects were drawn from a normal distribution. For every animal, the QTL genotype is available and thus the true breeding value (TBV) is nothing else but the sum of the QTL effects. Table 2 presents the Pearson correlation between the EBVs and TBVs for different sizes of the population in the dataset and different numbers of SNP markers used. Unfortunately, generating a dataset of 1,000,000 individuals genotyped for 360,000 SNPs was not feasible, due to limitations of the simulation software AlphaDrop. Although the Stevin computing infrastructure includes a machine with about 120 GB of
Table 1: Computational demands of DAIRRy-BLUP for different numbers of SNP markers included in the analysis for a population of 100,000 individuals.

<table>
<thead>
<tr>
<th>SNPs</th>
<th>Parallel processes</th>
<th>Wall time (hh:mm:ss)</th>
<th>Number of iterations</th>
<th>RAM per process (MB)</th>
<th>Total RAM (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9,000</td>
<td>16</td>
<td>00:10:56</td>
<td>3</td>
<td>93</td>
<td>1.5</td>
</tr>
<tr>
<td>60,000</td>
<td>32</td>
<td>02:54:48</td>
<td>2</td>
<td>608</td>
<td>19.5</td>
</tr>
<tr>
<td>360,000</td>
<td>720</td>
<td>17:48:10</td>
<td>2</td>
<td>1,154</td>
<td>831</td>
</tr>
</tbody>
</table>

working memory, it seemed that this was not sufficient to generate this large dataset. Nonetheless, DAIRRy-BLUP would be able to analyze such a dataset as only the read-in time would be ten times higher compared to the analysis of the dataset consisting of 100,000 individuals genotyped for 360,000 SNPs.

In the second column of Table 2, the results are listed for EBVs based on the QTL genotypes. In this case the random effects are the QTL of which it is known that they influence the phenotypic score by a predetermined effect. In the next paragraph, the estimates of the QTL effects are discussed, but here only the EBVs based on the estimated QTL effects are examined. As expected, it is confirmed that when all loci that contribute to the trait are known and genotypes for these loci are available, EBVs are always more accurate than when relying on genotypes for random SNPs. However, even when the exact positions of the QTL are known, it is observed that predictions of the EBVs are more accurate when the number of individuals is of the order of magnitude higher than the number of QTL. The results also indicate that it is more opportune to collect genotypic and phenotypic records of more individuals than to increase the number of SNP markers in the genotypic records, at least when the number of markers is already sufficiently higher than the number of QTL involved. Current large-scale genomic evaluations are performed for at most 50,000 individuals genotyped for around 50,000 SNPs (Tsuruta et al. 2013; Gao et al. 2012), where more information is included using sparse pedigree information of non-genotyped individuals. To improve prediction accuracies of these genomic

Table 2: Pearson correlation between the Estimated Breeding Values (EBVs) and the True Breeding Values (TBVs), for datasets with different numbers of individuals and genotyped for different numbers of SNPs. Column 2 displays the Pearson correlation between TBVs and EBVs, based on the estimated QTL effects (see Table 3). The missing value in the lower right corner is due to the impossibility to create such a dataset with AlphaDrop.

<table>
<thead>
<tr>
<th>Individuals</th>
<th>QTL</th>
<th>SNPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>9,000</td>
<td>0.768</td>
<td>0.643</td>
</tr>
<tr>
<td>100,000</td>
<td>0.939</td>
<td>0.610</td>
</tr>
<tr>
<td>1,000,000</td>
<td>0.99</td>
<td>0.612</td>
</tr>
</tbody>
</table>

evaluations, it is shown here that instead of using denser SNP arrays, it is more interesting to increase the number of genotyped individuals.

Estimating QTL effects

Breeding values already provide a lot of information to breeders, but it is sometimes interesting to have a good estimate of the marker effects. In the simulated datasets the phenotypic scores are determined by the true QTL effects and the QTL genotypes of the individuals. True SNP marker effects are not known because SNP markers are randomly drawn and there is no information available about the linkage disequilibrium between the QTL and the SNP markers. Therefore, only the Pearson correlation between the estimated QTL effects and the true QTL effects is given in Table 3. When comparing the results of Table 3 with those in Table 2, it is seen that although the estimates of the marker effects can be quite poor, the resulting estimates of the breeding values might still be fairly accurate. Of course, when trying to identify the positions of important QTL,
Table 3: Pearson correlation between the estimated QTL effects and the true QTL effects.

<table>
<thead>
<tr>
<th>Individuals</th>
<th>Prediction accuracy of QTL effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>0.354</td>
</tr>
<tr>
<td>100,000</td>
<td>0.669</td>
</tr>
<tr>
<td>1,000,000</td>
<td>0.851</td>
</tr>
</tbody>
</table>

it is essential to have access to accurate estimates of SNP marker effects and it is observed that large numbers of genotypic and phenotypic records are then required.

Parallel efficiency

The primary motivation for the development of DAIRRy-BLUP is that it enables the analysis of large-scale datasets beyond the memory capacity of a single node. This is achieved by the distribution of all matrices and vectors involved in the model among the local memories of a large number of computing nodes. Additionally, DAIRRy-BLUP leads to a significant reduction in runtime because the required computations are performed by several CPU cores concurrently. The parallel speedup $S_p$ is defined as $S_p = T_1/T_p$, where $T_p$ denotes the runtime on $P$ CPU cores. The parallel efficiency $\eta_p$ is defined as the ratio of the parallel speedup and the number of CPU cores used: $\eta_p = S_p/P$. In the ideal case, the speedup $S_p$ is equal to the number of cores used and the parallel efficiency is 100%. However, due to all sorts of overhead such as inter-process communication and load misbalance, the parallel efficiency is typically lower.

In order to determine the parallel efficiency $\eta_p$ for the largest problem, i.e., the dataset with 100,000 individuals genotyped for 360,000 SNPs, the runtime $T_1$ on a single CPU core is required. As it is impractical to obtain this value through a direct measurement, it is estimated through extrapolation of the runtime of a smaller problem with 100,000 individuals genotyped for 30,000 SNPs, taking into account the computational complexity of the algorithm. The dominating complexity is determined by the Cholesky decomposition and solution of the MME which are known to have cubic complexity. However, the construction of the coefficient matrix $C$ is also a time-consuming factor since it has a complexity of $O(ns^2)$ and the number of individuals ($n$) thus plays an important role when it has the same order of magnitude as the number of SNPs ($s$), as was the case for both datasets ($n = 100,000$). Therefore, the runtime for the construction of the coefficient matrix was extrapolated separately because of its quadratic complexity in the number of SNPs, as opposed to the cubic complexity of the other time-consuming parts of the algorithm.

Table 4 lists the computational demands for the smaller dataset with 30,000 SNPs when evaluated on a single CPU core and the estimated runtime and memory requirements for the large dataset with 360,000 SNPs on a single CPU core. The timings were averaged over the required iterations, because convergence was reached in a different number of iterations for both datasets. It is clear from this table that the analysis of the 360,000 SNPs dataset becomes computationally impractical when not applying any parallel programming paradigm as the processing of the dataset would require several months of computing time. By applying DAIRRy-BLUP on a cluster with 720 CPU cores, predictions of marker effects could be calculated roughly 400 times faster. This corresponds to a parallel efficiency in excess of 55%, which is generally considered acceptable. Additionally, the memory requirements do not pose any problem, since the distributed-memory framework allows for the aggregation of the local memory (2 GB) available to every CPU core, which means that a total of 1,440 GB RAM could be used.
Table 4: Illustration of the parallel efficiency of DAIRRy-BLUP for a dataset with 100,000 individuals genotyped for 360,000 SNPs. Speedup is calculated as the ratio between the computing time on a single CPU core and the computing time on \( P = 720 \) CPU cores. Parallel efficiency is defined as the ratio between the actual speedup and the maximum speedup, which is equal to the number of dedicated CPU cores.

<table>
<thead>
<tr>
<th>SNPs</th>
<th>Dedicated CPU cores ((P))</th>
<th>Time per iteration</th>
<th>Time for reading + set-up (\mathbf{C})</th>
<th>Time for solving (\text{MME})</th>
<th>Total memory usage, GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>30,000</td>
<td>1</td>
<td>6 h 44 min</td>
<td>5 h 03 min</td>
<td>1 h 41 min</td>
<td>3.73</td>
</tr>
<tr>
<td>360,000</td>
<td>1</td>
<td>151 d</td>
<td>30 d</td>
<td>121 d</td>
<td>537</td>
</tr>
<tr>
<td>360,000</td>
<td>720</td>
<td>8 h 50 min</td>
<td>1 h 49 min</td>
<td>7 h 00 min</td>
<td>831</td>
</tr>
</tbody>
</table>

\(\text{Speedup} \quad S_P = \frac{T_1}{T_N}\)

\(\text{Efficiency} \quad \eta_P = \frac{S_P}{P}\)

\(a\) Results for a dataset with 360,000 SNPs on a single CPU core are estimated based on results of the dataset with 30,000 SNPs, cubic complexity of the solving algorithm and square complexity of the set-up of matrix \(\mathbf{C}\) and memory usage.

DISCUSSION

As has been pointed out by Cole et al. (2012), there is a need for being able to process and analyze large-scale datasets using high performance computing techniques. DAIRRy-BLUP, a distributed-memory computing framework for genomic prediction is presented and meets some of the issues addressed by Cole et al. It has been shown that DAIRRy-BLUP is able to analyze large-scale genomic datasets in an efficient way when sufficient computing power is available. It is also suggested that analyzing such large-scale genomic datasets is necessary for obtaining better prediction accuracies for marker effects and breeding values, as was already suggested by Hayes et al. (2009) and Van-Raden et al. (2009). The results show that genotyping more individuals has a stronger effect on prediction accuracy than genotyping for a higher number of SNPs, although an increasing number of SNPs can also provide a higher prediction accuracy. This effect is in accordance with the observations by Habier et al. (2013), where a plateau was already reached for around 15,000 SNPs, but the phenotypes were only influenced by 200 QTL as opposed to 9,000 QTL in our study. The number of individuals was also quite low (maximum 2,000), but an increase of prediction accuracy was already noticeable when increasing the number of individuals in the training data. These results, together with the results obtained with DAIRRy-BLUP, justify the choice for an algorithm whose computational demands are dominated by the number of estimated random effects rather than by the number of individuals included in the analysis.

If the exact positions of the QTL on the chromosomes are known and animals can be genotyped for these QTL, prediction accuracies might improve substantially. Additionally, if a large number of phenotypic records of QTL-genotyped individuals is present, the effects of the different QTL can be estimated with significant accuracy. Current real datasets are probably still too small to obtain an accurate estimation of the marker effects; nonetheless, as the cost of genotyping is constantly decreasing, future datasets will contain the potential for identifying QTL on a large scale.

The results of this study on simulated datasets indicate that for accurate genomic predictions of breeding values, the number of genotyped individuals included in the analysis should increase substantially. Due
to the constant decrease in cost for genotyping animals or plants, it is assumed that such large-scale datasets will be available in the near future. Since analysis of these datasets on a single computing node becomes computationally impractical, DAIRRy-BLUP was developed to enable the application of a distributed-memory compute cluster for predicting EBVs based on the entire dataset with a significant speedup.

**ACKNOWLEDGEMENTS**

This study was funded by the Ghent University Multidisciplinary Research Partnership ‘Bioinformatics: from nucleotides to networks’ (N2N). The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by Ghent University, the Hercules Foundation and the Flemish Government – department EWI.

**LITERATURE CITED**


APPENDIX

A detailed derivation of the AI update matrix

Newton’s method for maximizing a likelihood function uses the Hessian matrix to update the variance components at each iteration (Eq. (2)). The elements of this matrix are the second order derivatives of the REML log likelihood with respect to the variance components:

\[
\frac{\partial^2 I_{\text{REML}}}{\partial \sigma^2} = -\frac{n-t}{2\sigma_e^2} \frac{y^T P_y}{\sigma^2_e}
\]

\[
\frac{\partial^2 I_{\text{REML}}}{\partial \sigma^2 \partial \gamma} = -\frac{y^T P_V P_y}{2\sigma^2_e}
\]

\[
\frac{\partial^2 I_{\text{REML}}}{\partial \gamma^2} = \frac{\text{tr}\left((P_V)^2\right)}{2} - \frac{\text{tr}(P_V)}{2}
\]

\[
= -\frac{y^T (P_V)^2 P_y}{\sigma^2_e} + \frac{y^T P_V P_y}{2\sigma^2_e}
\]

with \( V = \frac{\partial V}{\partial \gamma} \) and \( \gamma = \frac{\partial^2 V}{\partial \gamma^2} \). An alternative is to use the Fisher information matrix, whose elements are the expected values of the Hessian matrix:

\[
E\left(\frac{\partial^2 I_{\text{REML}}}{\partial \sigma^2}\right) = -\frac{n-t}{2\sigma^2_e}
\]

\[
E\left(\frac{\partial^2 I_{\text{REML}}}{\partial \sigma^2 \partial \gamma}\right) = -\frac{\text{tr}(P_V)}{2\sigma^2_e}
\]

\[
E\left(\frac{\partial^2 I_{\text{REML}}}{\partial \gamma^2}\right) = \frac{\text{tr}\left((P_V)^2\right)}{2}
\]

The elements of both matrices require the evaluation of traces which are very computer intensive. Therefore, it was suggested to use a simplified average of both matrices (Gilmour et al. 1995), called the average information matrix (\( H_{\text{AI}} \)), where the evaluation of those traces is no longer necessary:

\[
H_{\text{AI}}(\sigma^2_e, \gamma) = \frac{-y^T P_y}{2\sigma^2_e}
\]

\[
H_{\text{AI}}(\sigma^2_e, \gamma) = \frac{-y^T P_V P_y}{2\sigma^2_e}
\]

\[
H_{\text{AI}}(\gamma, \gamma) = \frac{-y^T (P_V)^2 P_y}{2\sigma^2_e}
\]

To obtain these equations, the expressions \( \frac{y^T P_V P_y}{\sigma^2_e} \) and \( \frac{y^T P_V P_y}{\sigma^2_e} \) are approximated by their expected values \( \text{tr}(P_V) \) and \( \text{tr}(P_V) \). When the variance structure is linear in \( \gamma \) and thus \( V = 0, H_{\text{AI}}(\gamma, \gamma) \) is the exact average of \( \frac{\partial^2 I_{\text{REML}}}{\partial \gamma^2} \) and \( E\left(\frac{\partial^2 I_{\text{REML}}}{\partial \gamma^2}\right) \).

Introducing a working variate \( y_{\gamma} = V_{\gamma} P y = \frac{Z\hat{u}}{\gamma} \), the AI-matrix can be calculated as:

\[
H_{\text{AI}} = \frac{1}{2\sigma^2_e} \left[ \begin{array}{c} y_{\gamma}^T \\ \gamma \end{array} \right] P \left[ \begin{array}{c} y_{\gamma} \\ \gamma \end{array} \right]
\]

Such a matrix \( Q^T P Q \) is also the Schur complement of the coefficient matrix \( C \) in matrix \( M \):

\[
M = \begin{bmatrix} Q^T R^{-1} Q & Q^T R^{-1} W \\ W^T R^{-1} Q & C \end{bmatrix}
\]

\[
W = \begin{bmatrix} X & Z \end{bmatrix}
\]

(A.1)

Algorithmic details: distribution of matrices among all processes

The distributed memory computing approach involves splitting up large matrices into smaller blocks that are distributed among the available processes. A process
Figure 1: Schematic representation of a 6 by 11 matrix distributed among 4 processes using 2D block cyclic distribution with blocks of size 2 by 2. P is the process that holds into memory the elements of the matrix, inside the small blocks delineated by the dashed lines.

is an instance of DAIRRy-BLUP that is executed on a CPU core of one of the available networked nodes. In this way every process can perform certain operations in parallel on the blocks that are at its disposal. When a certain process needs information about a block that is assigned to another process, it can receive this information through message passing over the network. To obtain maximal efficiency, it is advised to map every process on a one-to-one basis onto the available CPU cores and to set the number of processes such that neither the available amount of working memory per CPU core is superseded nor that communication between processes exceeds computation time.

PBLAS and ScaLAPACK are libraries that can perform most essential algebraic operations on distributed matrices. They rely on a 2D block cyclic distribution of matrices as is schematically depicted in Figure 1.

It is easily seen that addition of such a matrix with another matrix, distributed in the same way, can be performed in parallel on all processes without any communication overhead. The details of more complex operations performed on distributed matrices can be found in specialized literature (e.g. Choi et al. 1996b; Van De Geijn and Watts 1997).

In DAIRRy-BLUP all available processes only read in their parts of the matrices and as illustrated in Figure 1. The matrix multiplications needed for the set-up of the MME are performed in a distributed way and the resulting coefficient matrix is thus directly distributed among all available processes. The distribution of the coefficient matrix is not changed by any operation performed on the coefficient matrix.

**Algorithmic details: Calculation flow**

The calculation flow for the DAIRRy-BLUP implementation is presented here. The way how matrices are distributed among processes is not explicitly mentioned since this depends on the number of processes involved and the chosen size of the blocks. The processes are always mapped to a 2D grid where the number of columns is as close as possible to the number of rows, since this is recommended in the ScaLAPACK User’s Guide (Blackford et al. 1997). Below are listed the main steps of the algorithm; the complete code can be obtained at https://github.ugent.be/pages/ardconin/DAIRRy-BLUP/.

1. Every process reads in the blocks of matrices and that are assigned to it by the 2D block cyclic distribution and the coefficient matrix as well as the right-hand side of the MME are calculated using Eq.
(4) for obtaining $Z'Z$, $X'X$, $X'Z$, $Z'y$ and $X'y$. The first two multiplications are calculated using the level 3 PBLAS routine PDSYRK, the others using level 3 PBLAS routine PDGEMM.

2. Compute the Cholesky decomposition of $C = LL'$ in parallel using the PDPTORF ScalAPACK routine.

3. Estimates for the random and fixed effects are obtained by directly solving the MME using the Cholesky decomposition of $C$ and the PDPTORS ScalAPACK routine.

4. An estimation for $\sigma^2_e$ is calculated by all processes as (using $Q = y$ in Eq. (A.1)):

$$\sigma^2_e = \frac{y'y}{n-t} = \frac{1}{n-t}\left( \begin{bmatrix} y'\ y \\ W \\ b \end{bmatrix} ' \begin{bmatrix} y'\ y \\ W \\ b \end{bmatrix} \right),$$

where $y'y$ is calculated as the square of the Euclidian norm of $y$, obtained with the level 1 PBLAS routine PDNRM2.

5. Construct matrices $X'Q$ and $Z'Q$, with $Q$ as defined in Eq. (3), using level 3 PBLAS routine PDGEMM and solve the following linear equation with the PDPTORS ScalAPACK routine for matrix $R$:

$$CR = \begin{bmatrix} X'Q \\ Z'Q \end{bmatrix}$$

6. The AI update matrix is calculated as the Schur complement of the coefficient matrix inside the matrix $M$ (Eq. (A.1)), which can be calculated as:

$$H_{AI} = Q'yQ = Q'y - Q'yWR$$

7. Using the Cholesky decomposition the inverse of $C$ is calculated with the PDPTORI ScalAPACK routine and the trace of $C^{-1}$ is calculated with the BLACS routine DGSUM2D for obtaining the score function as defined in Eq. (1).

8. As long as the relative updates for $\sigma^2_u$ and the REML log-likelihood are not smaller than $\varepsilon = 0.01$, repeat from step 1 using an updated value for $\sigma^2_u$ and $\sigma^2_e$. The iterative cycle is also stopped when the maximum number of iterations is reached (default 20). The values of $\varepsilon$ and the maximum number of iterations can be changed by the user to obtain faster convergence or more accurate solutions.

9. Optionally, breeding values can be calculated for a large-scale dataset based on the estimates of the SNP marker effects. Genotypes for the testset are read in by all processes in a distributed way as defined by the 2D block cyclic distribution. To minimize memory requirements, breeding values are calculated in strips, which means that only a few breeding values are calculated in parallel by the level 3 PBLAS routine PDGEMM. The number of breeding values calculated in parallel is defined by the chosen blocksize and the number of dedicated processes.