Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Solving sequences of generalized least-squares problems on multi-threaded architectures

Diego Fabregat-Traver [a,*], Yurii S. Aulchenko [b], Paolo Bientinesi [a]

[a] Aachen Institute for Advanced Study in Computational Engineering Science, RWTH Aachen, 52062 Aachen, Germany
[b] Institute of Cytology and Genetics, SD RAS, 630090 Novosibirsk, Russia

ARTICLE INFO

Keywords:
Numerical linear algebra
Generalized least-squares
Sequences of problems
Shared-memory
Out-of-core

ABSTRACT

Generalized linear mixed-effects models in the context of genome-wide association studies (GWAS) represent a formidable computational challenge: the solution of millions of correlated generalized least-squares problems, and the processing of terabytes of data. We present high performance in-core and out-of-core shared-memory algorithms for GWAS: by taking advantage of domain-specific knowledge, exploiting multi-core parallelism, and handling data efficiently, our algorithms attain unequalled performance. When compared to GenABEL, one of the most widely used libraries for GWAS, on a 12-core processor we obtain 50-fold speedups. As a consequence, our routines enable genome studies of unprecedented size.

1. Introduction

Generalized linear mixed-effects models (GLMMs) are a type of statistical model widespread in many different disciplines such as genomics, econometrics, and social sciences [1–3]. Applications based on GLMMs face two computational challenges: the solution of a sequence comprising millions of generalized least-squares problems (GLSs), and the processing of data sets so large that they only fit in secondary storage devices. In this paper, we target the computation of GLMMs in the context of genome-wide association studies (GWAS).

GWAS is the tool of choice to analyze the relationship between DNA sequence variations and complex traits such as diabetes and coronary heart diseases [4–6]. More than 1300 papers published during the last five years endorse the relevance of GWAS [7]. The GLMM specific to GWAS computes

$$b_i := (X_i^T M^{-1} X_i)^{-1} X_i^T M^{-1} y, \quad \text{with } 1 \leqslant i \leqslant m, \tag{1}$$

where y is the vector of observations, representing a given trait or phenotype; X_i is the design matrix, including covariates and genome measurements; M represents dependencies among observations; and b_i represents the relation between a variation in the genome sequence (X_i) and a variation in the trait (y). In linear algebra terms, Eq. (1) solves a linear regression with non-independent outcomes where $b_i \in \mathcal{R}^p$, $X_i \in \mathcal{R}^{n\times p}$ is full rank, $M \in \mathcal{R}^{n\times n}$ is symmetric positive definite (SPD), and $y \in \mathcal{R}^n$; the sizes are as follows: $n \approx 10^4$, $2 \leqslant p \leqslant 20$, and m, the length of the sequence, ranges from 10^6 to 10^8. The quantities X_i, M, and y are known. Additionally, the X_i's present a special structure that will prove to be critical for performance: each X_i may be partitioned as $(X_L \mid X_{R_i})$, where $X_L \in \mathcal{R}^{n\times(p-1)}$ is the same for all X_i's, and $X_{R_i} \in \mathcal{R}^{n\times 1}$ varies.

* Corresponding author.
E-mail addresses: fabregat@aices.rwth-aachen.de (D. Fabregat-Traver), yurii.aulchenko@gmail.com (Y.S. Aulchenko), pauldj@aices.rwth-aachen.de (P. Bientinesi).

http://dx.doi.org/10.1016/j.amc.2014.02.056
0096-3003/© 2014 Elsevier Inc. All rights reserved.

1.1. Limitations

Computational biologists performing GWAS aim for the sizes described above; in a typical scenario with $n = 10,000, p = 4$, and $m = 36 \times 10^6$, 3 Terabytes of data have to be processed through 3.6×10^{15} arithmetic operations (Petaflops). In practice, current GWAS solvers are constrained to much smaller problems due to time limitations. For instance, in [8], the authors carry out a study that takes almost 4 h for the following problem sizes: $n = 1,500$, $p = 4$, and $m = 220,833$. The time to perform the same study for $m = 2.5$ million is estimated to be roughly 43 h. With our routines, the time to complete the latter reduces to 10 min.

1.2. Terminology

We collect and give a brief description of the acronyms used throughout the paper.

- GWAS: genome-wide association studies.
- GLS: generalized least-squares problems.
- GenABEL: one of the most widely used frameworks to perform GWAS.
- GWFGLS: GenABEL's state-of-the-art routine for the solution of Eq. (1).
- HP-GWAS: our novel in-core solver for Eq. (1).
- OOC-HP-GWAS: out-of-core version of HP-GWAS.

Table 1 enumerates the BLAS (Basic Linear Algebra Subprograms) [9] and LAPACK (Linear Algebra PACKage) [10] routines used in the algorithms presented in this paper. LAPACK and BLAS are the de facto standard libraries for high-performance dense linear algebra computations.

1.3. Related work

Traditionally, LAPACK is the tool of choice to develop high-performance algorithms and routines for linear algebra operations. Although LAPACK does not support the solution of a single GLS directly, it offers routines for closely related problems: GELS for least squares problems, and GGGLM for the general Gauss-Markov linear model. Algorithms 1 and 2 provide examples for the reduction of GLS problems to GELS and GGGLM, respectively. Unfortunately, none of the algorithms provided by LAPACK is able to exploit the sequence of GLSs within GWAS, nor the specific structure of its operands. Conversely, existing ad hoc routines for Eq. (1), such as the widely used GWFGLS, are aware of the specific knowledge arising from the application, but exploit it in a sub-optimal way.

LAPACK and GenABEL present additional drawbacks: LAPACK routines are in-core, i.e., data is expected to fit in main memory, otherwise the use of virtual memory results in a serious performance loss; since GWAS may involve terabytes of data, it is in general suboptimal to rely on these routines directly. Contrarily, GenABEL incorporates an out-of-core mechanism, but it suffers from significant overhead.

Algorithm 1. GLS problem reduced to GELS

```
1  $LL^T = M$            (POTRF)
2  $y := L^{-1}y$       (TRSV)
3  $X := L^{-1}X$       (TRSM)
4  $b := \text{GELS}(X, y)$
```

Algorithm 2. GLS problem reduced to GGGLM

```
1  $LL^T = M$            (POTRF)
2  $b := \text{GGGLM}(X, y, L)$
```

1.4. Contributions

We present high-performance in-core and out-of-core algorithms, HP-GWAS and OOC-HP-GWAS, and their corresponding routines for the computation of GWAS on multi-threaded architectures.

Our algorithms are optimized not for a single instance of the GLS problem but for the whole sequence of such problems. This is accomplished by

- breaking the black box structure of traditional libraries, which impose a separate routine call for each individual GLS,
- exploiting domain-specific knowledge such as the particular structure of the operands,