Creation of genomic relationship matrices and BLUPF90 “user file”

Ignacio Aguilar

INIA, Uruguay

05-2018

Genomic Relationship Matrix - G

- G = ZZ’/k
  - Z = matrix for SNP marker
  - Dimension Z= n*p
  - n animals,
  - p markers

Genotype Codes

0 – Homozygous
1 – Heterozygous
2 – Homozygous

5 – No Call (Missing)

Data file with SNP marker

```
80   211010110020120110110101101111112111112101 00
8014 211101015111011202211101115111121011122101 00
516  211001012022520211202101211021112022121111 01
181  211101111122011205502000201010222122111111 00
```

HOWTO: Creation of Genomic Matrix

- Read SNP marker information => M

$$\begin{bmatrix} 2 & 1 & 2 & .. \\ 0 & 1 & 0 & .. \\ .. & .. & .. & .. \end{bmatrix}$$

- Get 'means' to center
  - Calculate allele frequency from observed genotypes (p_i)
  - p_i= sum($SNPcode_i$)/2n

- Matrix for center W(3,p)

$$\begin{array}{c} 0 \\ 1 \\ 2 \end{array} \begin{bmatrix} 0-2p_1 & 0-2p_2 & .. \\ 1-2p_1 & 1-2p_2 & .. \\ 2-2p_1 & 2-2p_2 & . \end{bmatrix}$$

- Center matrix Z = W(M)

Creation of Genomic Relationship Matrix

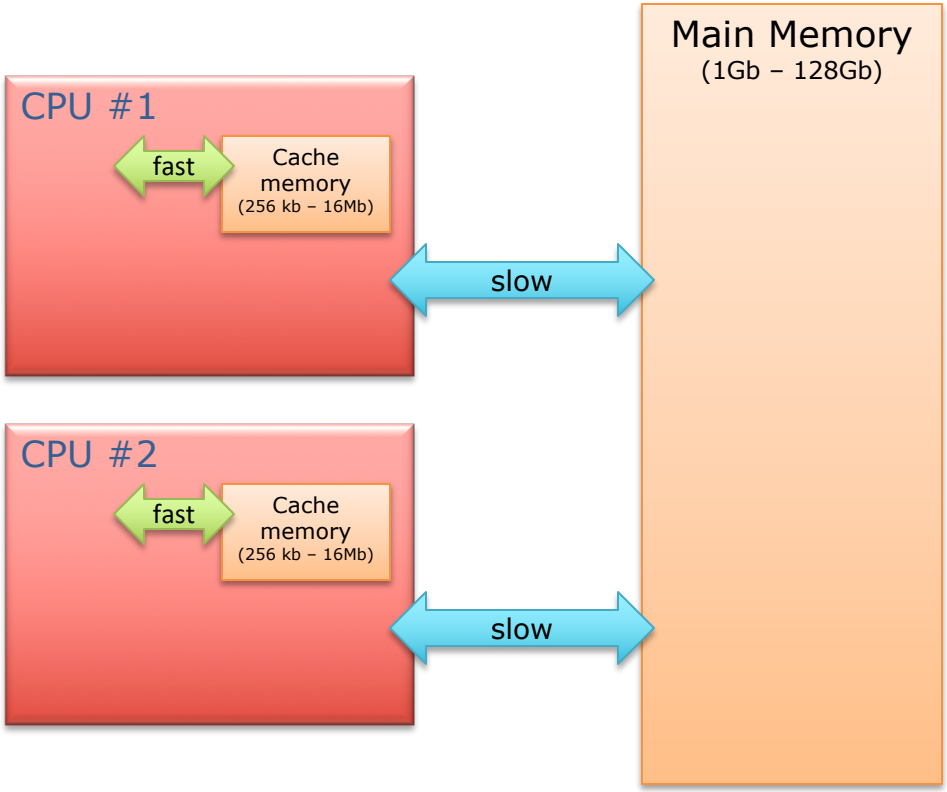
- Issues
  - Large number of genotyped individuals
  - Large number of SNP markers
  - Matrix multiplication ~ cost n^2 * p

- Large amount of data put in (cache) memory for doing ‘matmul’ for each pair of animals and indirect memory access (center)
  - Memory hierarchy

Matrix multiplication

- Matrix multiplication
  - Several methods
    - Intrisic matmul (good for small examples !!!)
    - “do-loops”
    - Packages (BLAS, LAPACK)
      - Non-optimzed
      - Optimized (ATLAS, MKL, etc.)
  - Several Compilers
    - Perform automatic optimization
      - Vectorize loops
      - Detect permuted loops
    - Can use OpenMP directives for parallelization

Memory Hierarchy



Alternative codes to create G matrix

Original

```
Do i=1,n
        Do j=i,n
                S=0
                Do k=1,p
                        S=S+Z(M(i,k),k)
                          *Z(M(j,k),k)
                End do
                G(i,j)=S/sqrt(d(i)*d(j))
                G(j,i)=G(i,j)
        End do
End do
```

Gmatrix.f90 (VanRaden, 2009)

Optimize Indirect Memory Access -OPTM

```
Do k=1,p
        X(:,k)=Z(M(:,k),k)
End do
Do i=1,n
        Do j=i,n
                S=0
                Do k=1,p
                        S=S+X(i,k)
                          *X(j,k)
                End do
                G(i,j)=S/sqrt(d(i)*d(j))
                G(j,i)=G(i,j)
        End do
End do
```

Optimize Memory and Loops - OPTML

```
Do k=1,p
        X(:,k)=Z(M(:,k),k)
End do
Do i=1,n
        Do j=1,n
                Do k=1,p
                        G(i,j)=G(i,j)
                          +X(i,k)*X(j,k)
                End do
        End do
End do
Do i=1,n
        Do j=1,n
                G(i,j)=G(i,j)/sqrt(d(i)*d(j)
        End do
End do
```

CPU time for alternative codes for G matrix and machines

- Testing
  - 6500 genotyped animals
  - 40k SNPs

		Algorithms		
Processor	**Cache**	**Original**	**OPTM**	**OPTML**
Xeon 3.5 GHz	6 MB	24 m	26 m	7 m
Opteron 3.0 GHz	1 MB	265 m	59 m	17 m

Aguilar et al 2011

CPU time (m) with alternative codes and compilers

- Testing
  - 6500 genotyped animals
  - 40k SNPs
  - Opteron 3.02 GHz 1 MB Cache memory

Compiler	Original	OPTM	OPTML
Intel	265	59	17
Absoft	241	60	34
gfortran	213	63	>1day

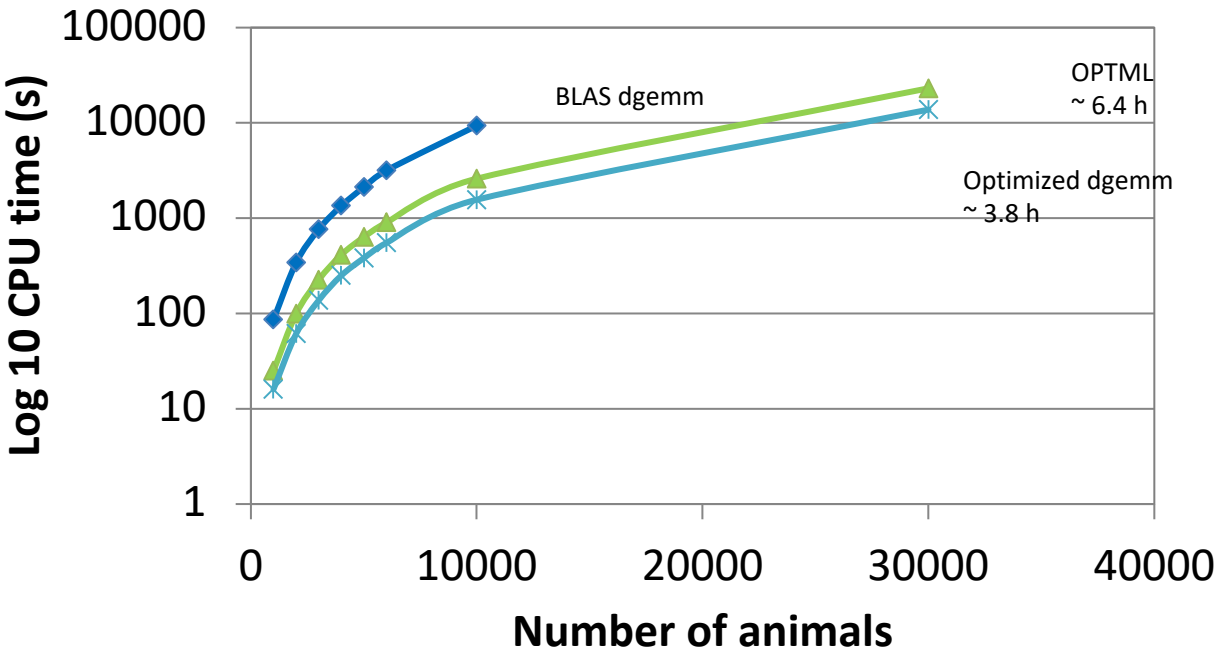
Aguilar et al 2011

Matrix multiplication subroutines

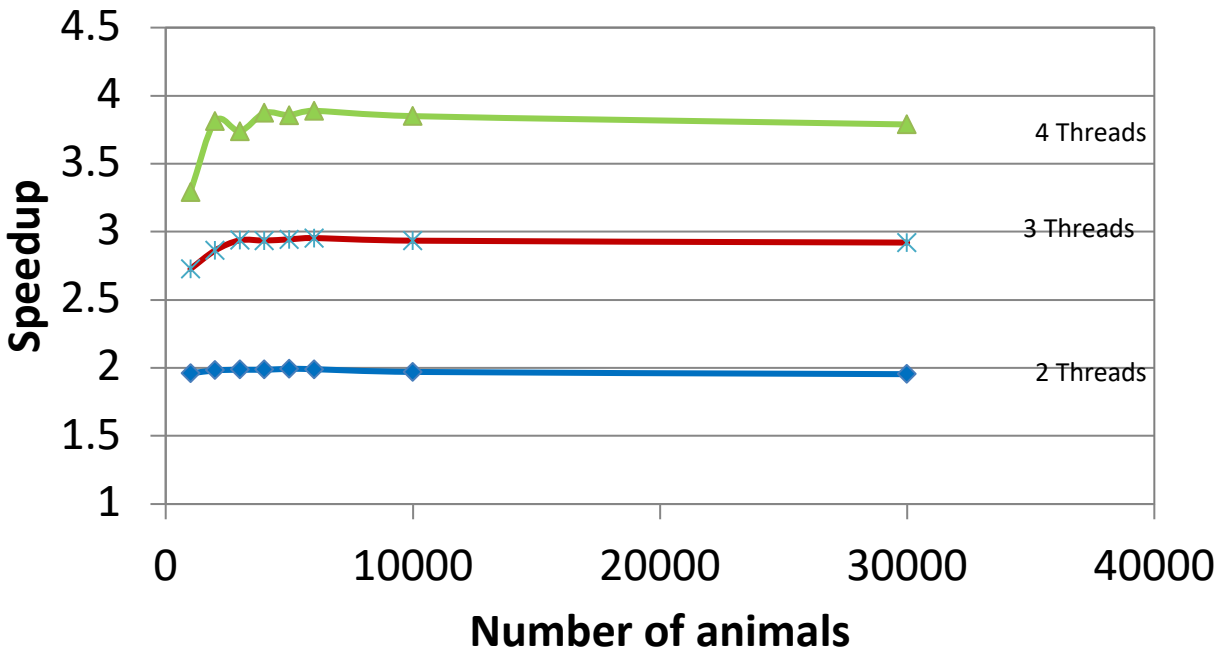
- Optimized memory and loops (compiler optimization)
- *dgemm* subroutine from BLAS
- Optimized *dgemm* (ATLAS or MKL libraries*)
  - Serial
  - Parallel (Automatic use of OpenMP)

  * Intel Fortran Compiler

Matrix multiplication using 40k SNPs



Speedup for matrix multiplications



Speedup = time using one thread/time using *n* threads

Efficent methods to construct genomic relationship matrices

Elapsed time for different number of individuals
BLADE INIALB 24 cpu

Number of genotypes	Genomic Relationship Matrix	
	Creation	Invertion
10k	0.6 m	0.1 m
30k	5.4 m	3 m
50k	15 m	14 m
70k	30 m	36.4 m
100k	60 m	106 m

`PreGSf90` program

- Creation and handling of genomic relationship matrices and relationship based on pedigree
- Different methods to optimize calculations using parallel processing
- Genomic module from BLUPF90 package

BLUPF90 parameter file

```
# Each keyword can be preceded by comments, each starting with
#
DATAFILE
name of data file
NUMBER_OF_TRAITS
number of traits
NUMBER_OF_EFFECTS
number of effects
OBSERVATION(S)
position of observations in data file (one per trait)
WEIGHT(S)
position of weight(s) in data file (one per trait); blank if
all weights equal
EFFECTS: POSITIONS_IN_DATAFILE NUMBER_OF_LEVELS TYPE_OF_EFFECT
[EFFECT NESTED]
one line per effect:
     position of effect (one per trait)
     number of levels
     type of effect (one of cross for crossclassified or cov
for covariable)
     position of effect where nested (one per trait, optional)

RANDOM_RESIDUAL VALUES
residual variance covariance matrix (full stored)
RANDOM_GROUP
number of one random effect or list of correlated effects
(order as in EFFECTS above); correlated effects must be
consecutive
RANDOM_TYPE
type of random effect:   one of diagonal, add_animal,add_sire,
                         add_an_upg, add_an_upginb, user_file,
                         user_file_i or par_domin
FILE
relationship file; blank line if file unnecessary
(CO)VARIANCES
variance covariance matrix for given correlated effects and
traits(full stored)
```

Repeat for each Random effect

Random effect definition

- RANDOM_GROUP
  - Number(s) of effect from list of effects
  - Correlated effects should be consecutive e.g. Maternal effects, Random Regression models
- RANDOM_TYPE
  - diagonal, add_animal, add_sire, add_an_upg, add_an_upginb, **user_file**, user_file_i or par_domin
- FILE
  - Pedigree file, parental dominance or user file
- (CO)VARIANCES
  - Square matrix with dimension equal to number_of_traits*number_of_correlated_effects

user_file Random Type in BLUPF90

- If (co)variance structure not supported by default BLUPF90 random type (i.e. diagonal, animal additive genetic, etc.)
- Allows to use a predefined (co)variance structure created by the user outside of BLUPf90
- (co)variance structures will be included in the mixed model equations for the corresponding random effect
- Supported in all programs: BLUPF90, (AI)REMLF90, GIBSSxF90
- Not yet supported by RENUMF90

user_file examples

- Plant breeding (with tetra or hexa-ploids or self-fertilization)
- Uncertain paternity
- Kernel methods such as RKHS
- Autoregressive covariance structures
- Genomic relationship matrices
- Genomic dominance
- etc. …

user_file definition in BLUPF90 parameter file

- Include a random effect of the type ”user_file”
- Provide the name of file that contains the ***inverse*** of the covariance structure

```
RANDOM_GROUP
 # genomic
 2
RANDOM_TYPE
user_file
FILE
 # matrix file
Gi
```

File for random effect “user_file”

File with following form:

- 3 Columns: row, col, value
  - Row/Col corresponds to the levels of the random effect
- stored as lower- or upper- triangle
- Zero values could be omitted
- plain text delimited with space

Original matrix

```
1.0  0.0  0.5
0.0  0.5  0.0
0.5  0.0  1.1
```

Upper-triangle matrix

```
1 1 1.0
1 3 0.5
2 2 0.5
3 3 1.1
```

Example of file for random effect “user_file”

```
1 1 .999382118619
1 2 .355052761478
2 2 1.014521277458
1 3 -.048184197960
2 3 -.057513012886
3 3 .976558921904
1 4 -.101734083083
2 4 -.007644724611
3 4 .196757165096
4 4 1.018165021903
```

BLUPF90 parameter file

- 3 effects
- 1 fixed effect (general mean)
- 2 user_file random effects defined

```
DATAFILE
 pheno.dat
NUMBER_OF_TRAITS
     1
NUMBER_OF_EFFECTS
     3
OBSERVATION(S)
     4
WEIGHT(S)

EFFECTS: POSITIONS_IN_DATAFILE NUMBER_OF_L
  3       1 cross
  1    3534 cross
  1    3534 cross
RANDOM_RESIDUAL VALUES
   1.1750
RANDOM_GROUP
  2
RANDOM_TYPE
user_file
FILE
snp.dat.Gi
(CO)VARIANCES
  0.50
RANDOM_GROUP
  3
RANDOM_TYPE
user_file
FILE
snp.dat.DDi
(CO)VARIANCES
  0.285
```