Adjacency-constrained hierarchical clustering of a similarity matrix

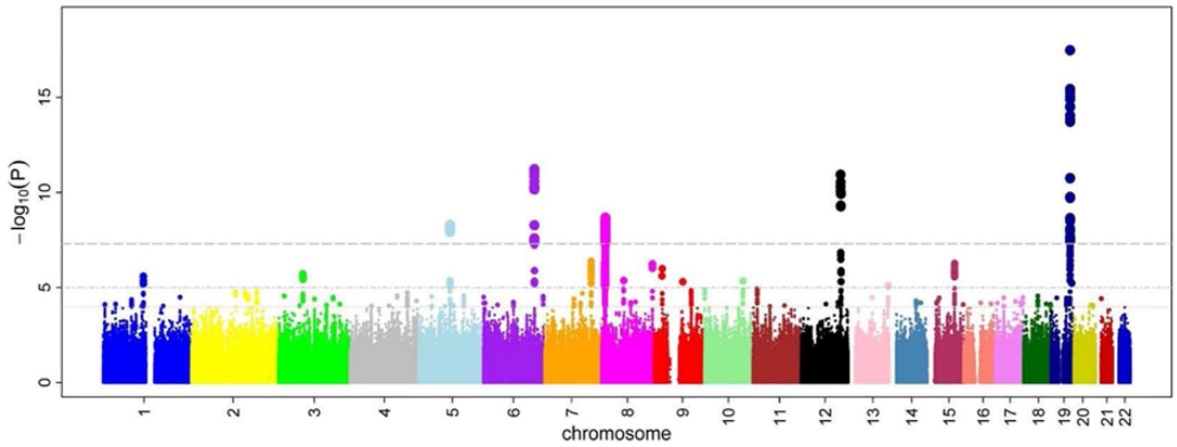
C. Ambroise, A. Dehman, M. Koskas, **P. Neuvial**, G. Rigaill

Rencontres R 2016, Toulouse

Context: linkage disequilibrium in GWAS

Genome-Wide Association Studies

- Goal: identify genetic markers (SNP) associated with a phenotype (disease) of interest.
- State-of-the art approach: Univariate tests of association between each of p markers and the phenotype.

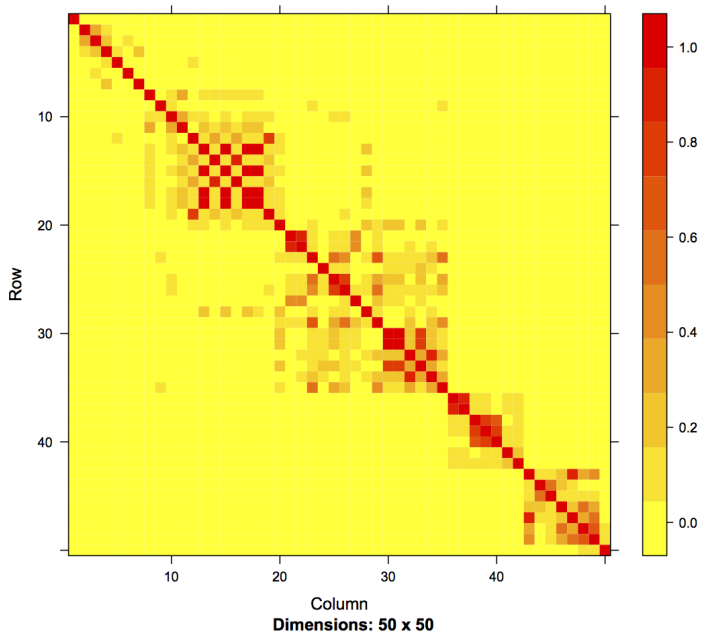


- $p \sim 10^6$ tests (genomic markers)
- statistical dependency between markers

(Image from: Ikram MK *et al.*, *PLoS Gen.* 2010)

Linkage disequilibrium (LD)

- LD: the non-random association of alleles at two or more loci
- May be quantified by $r^2(j, j') = \text{corr}(Z_j, Z_{j'})$, where $Z_j = \mathbf{1}_{\text{minor allele is present for SNP } j}$
- LD is structured in **diagonal blocks**:



r^2 coefficients for the first 50 SNP on chromosome 22 in Dalmasso *et al.*, *PLoS One*, 2008

This dependency is generally not accounted for in GWAS.

A block-wise approach for GWAS

Goal: detect **blocks of adjacent SNPs** associated with phenotype

A three-step approach[1]

1. Hierarchical clustering of the SNPs with adjacency constraint using the LD similarity
2. Estimation of the optimal number of groups: Gap statistic
3. Selection of the associated blocks: Group Lasso regression

Implemented in package `BALD`

Computational bottleneck

clustering step: steps 1. . . and 2!

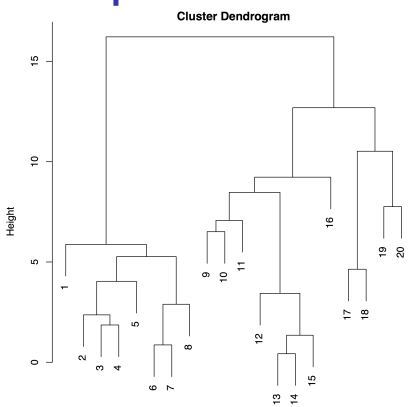
[1] Dehman, Ambroise, Neuvial *BMC Bioinformatics* 2015

Adjacency-constrained clustering

Inputs

- data matrix: genotypes of p SNPs
- similarity measure between features: LD between SNPs
- similarity measure between clusters of features: Ward

Output: a dendrogram



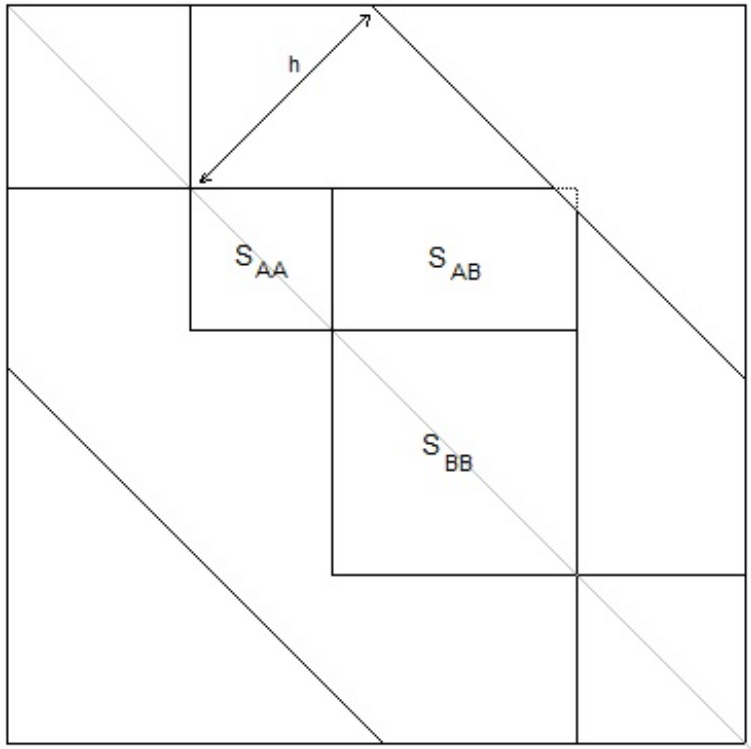
Algorithm

Starting from p clusters of individual features, iteratively merge of the two nearest clusters **among adjacent clusters**

Ward's distance between classes A and B

$$d(A,B) = \frac{p_A p_B}{p_A + p_B}\left(p_A^{-2} S_{AA} + p_B^{-2} S_{BB} - 2 p_A^{-1} p_B^{-1} S_{AB}\right),$$

where $S_{IJ} = \sum_{i \in I, j \in J} \mathrm{LD}(i,j)$.



d(A, B) only depends of sums of LD within blocks

Time and space complexities

Package	Method	time	space
`hclust`	Without adjacency constraint	$O(p^3)$	$O(p^2)$
`rioja`	Adj. c. with pre-calculation of LD	$O(p^2)$	$O(p^2)$
`BALD`	Adj. c. w/o pre-calculation of LD	$O(p^2)$	$O(p)$

Improving space complexity

(at no price on time complexity)

- Adjacency constraint: from $O(p^3)$ to $O(p^2)$
- Without *pre-calculation* of all pairwise LD: from $O(p^2)$ to $O(p)$

Lower bound on time complexity: $O(p^2)$

- All pairwise LD need to be calculated.

Numerical experiments

HIV data

- Dalmasso *et al.*, *PLoS One*, 2008
- 605 individuals, 300 000 SNP

Execution time for *one* run of the algorithm

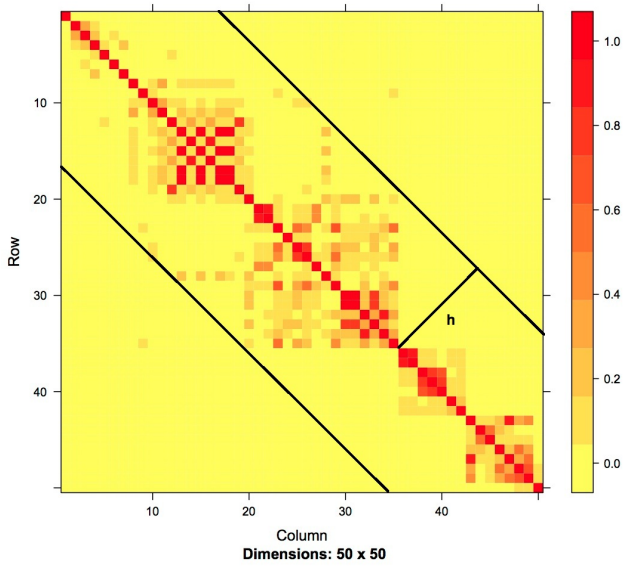
chromosome	22	1	whole genome
size	5417	23304	307851
time (hours)	0.16	3	538

Can we do faster?

A faster, approximate solution

A simplifying biological assumption

LD induces **short range** dependency compared to p:



- $\mathrm{LD}(j, j') \sim 0$ when $|j' - j| > h$
- a $p \times h$ similarity matrix, with $h \ll p$

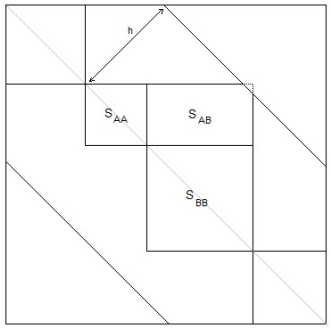
Only $O(ph)$ ***pairwise values of LD need to be calculated***

Are we done yet ?

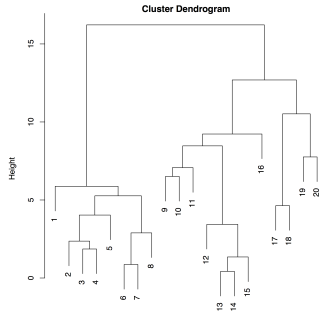
LD coefficients can be pre-calculated and stored in memory ($p \times h$)
. . . but the algorithm is still quadratic in p (in time)!

Two remaining challenges

p successive merging steps to be performed. For each of them:



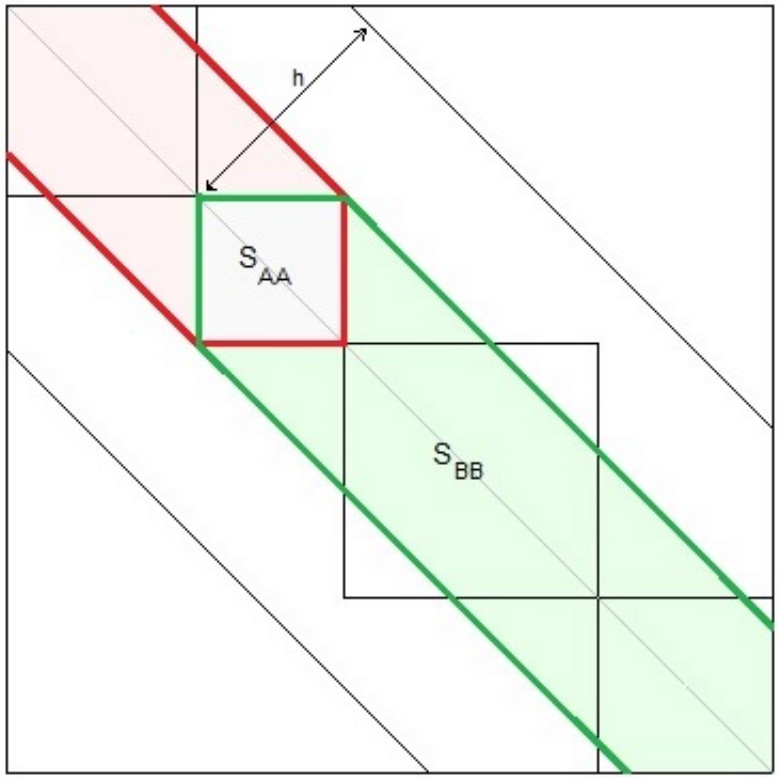
1. sums of the form S_{IJ} must be calculated smartly

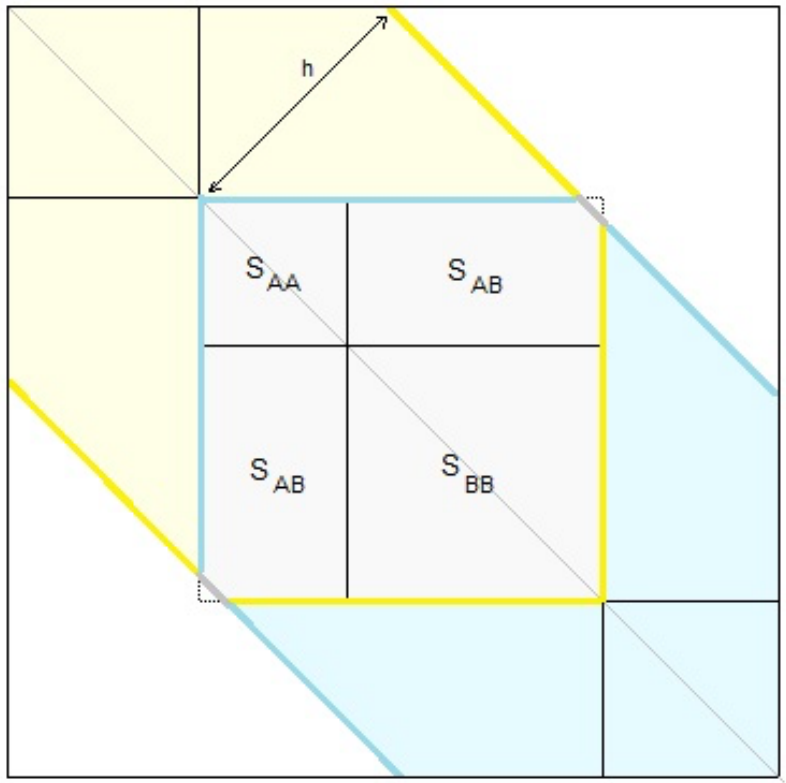


finding the next best merge is linear in p
2. need an efficient way (ie $o(p)$) to store and retrieve candidate merges

1- Pre-calculation of within-block sums

(a.k.a. the "pencil trick")

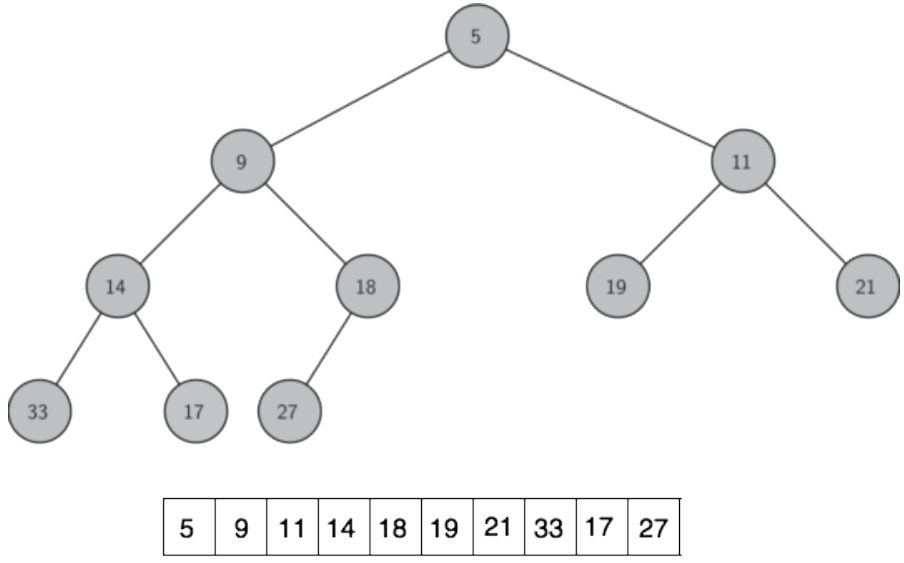




$\Rightarrow$ Two arrays of sizes $p \times h$ for storing the pencil sums.

2- Binary heaps to store merge candidates

- Each parent is **smaller** than its children
- Given a position i:
  - Parent$(i) = \lfloor i/2 \rfloor$
  - Left$(i) = 2i$
  - Right$(i) = 2i + 1$



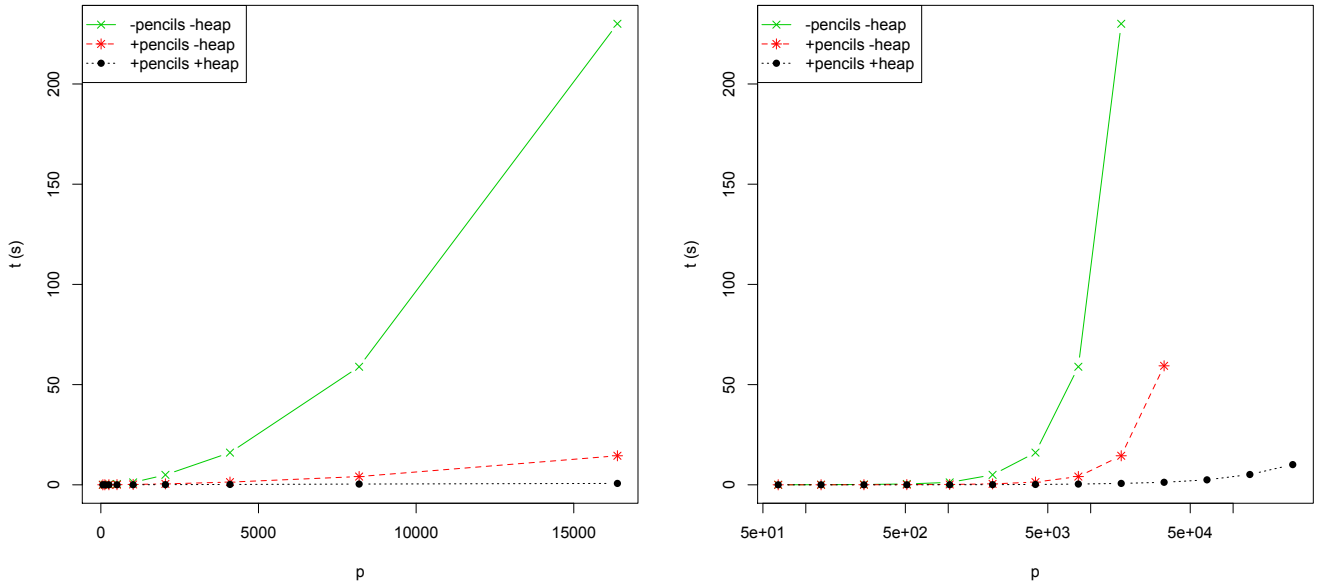
storage	retrieve min	insert	delete min
unordered array	$O(p)$	$O(1)$	$O(p)$
min-heap	$O(1)$	$O(\log(p))$	$O(\log(p))$

A quasilinear time implementation

1. band similarity matrix
2. pre-calculation of specific cumulative sums of LD
3. storing candidate merges using a binary heap
4. implementation of the binary heap in C

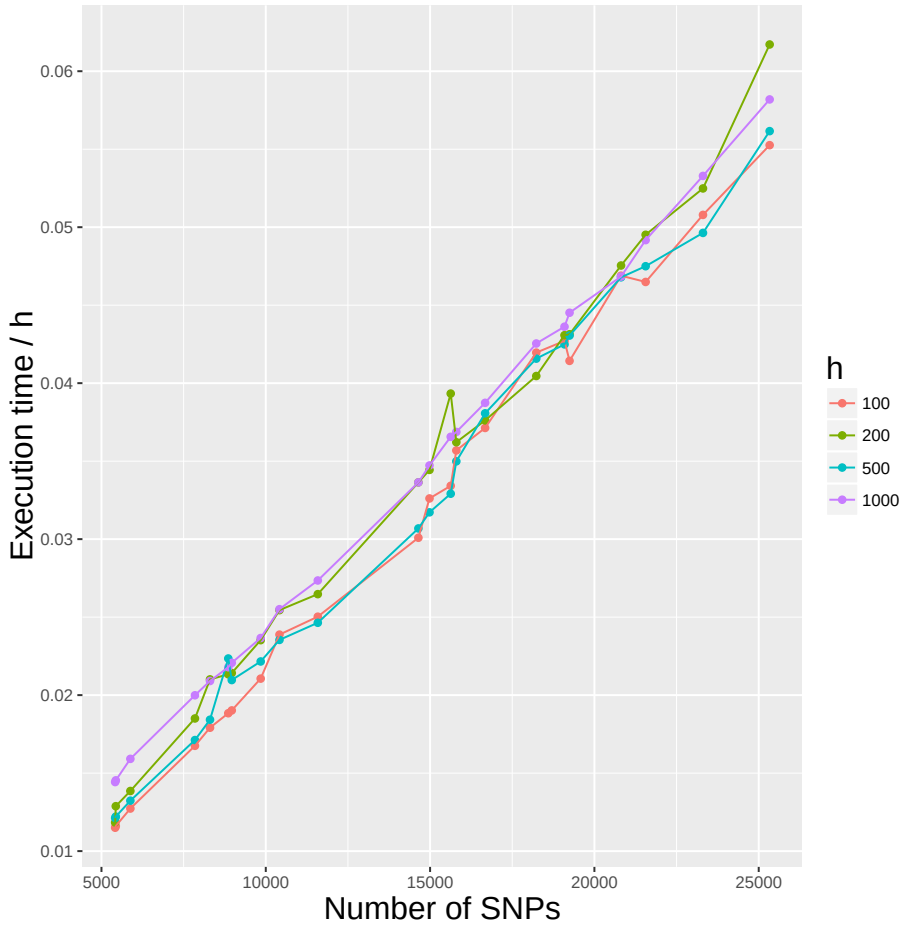
Time and space complexity: $O\left(p(\log(p) + h)\right)$

Numerical experiments on synthetic data



Numerical experiments on real data

Linear execution time in p and h on real data



Conclusion and perspective

Summary

- Incorporating biological constraints can be useful
- Good tradeoff between time and space complexity

Perspective

- p-values on blocks (Suzuki & Shimodaira, package `pvclust`)
- Application to other problems (Hi-C ?)